UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Master thesis

Using Auto-Regression to Predict Molecular Forces

Birk N. Dissing

Prof. Gemma C. Solomon

Handed in: October 31, 2024

Abstract

Ab initio molecular dynamics is a way of studying the dynamics of a chemical system by evaluating the forces on the atoms at each time step using quantum mechanical methods. The main advantages of *ab initio* molecular dynamics over other molecular dynamics are its ability to be applied to many different systems and handle changes in the electronic configuration. However, using quantum mechanical methods to evaluate forces is computationally heavy, limiting the timescale of simulations to picoseconds. There is great interest in speeding up *ab initio* molecular dynamics, as that will both lead to faster simulations and the ability to explore phenomena at longer timescales. In this thesis, I created an auto-regressive model that uses force trajectories to forecast the forces on atoms without needing to be fit to the system using a large training set that is time-consuming to generate. When my model was used to simulate ethanol, it did not perform better than simply increasing the time step of the simulation to achieve an equal speed-up. My model performed better on heavier molecules, yielding valid solutions to aspirin and cholesterol simulations. However, even in the best-case scenarios, my model only performed slightly better than increasing the time step of the simulations to achieve the same speed-up. While my model does not generally perform better than simply increasing the time step, the applied model and input features are simple. It is possible that time series forecasting can yield competitive results with a more advanced model and input features.

Contents

1 Introduction

With the rise of computational power and methods in the last decades, molecular dynamics (MD) have become an increasingly important tool in the modern chemist's toolbox. MD is a method that can simulate the physical movements of atoms and molecules by estimating the forces acting on the atoms and solving Newton's laws of motion. MD can be used to explore how atoms and molecules interact with each other over time, allowing us to explore the dynamics of different systems. When doing MD one of the main problems is how to best approximate the interactions between atoms in a system, as it is too complicated to solve completely.[1–3] Different MD methods have different ways of approximating this problem. Classical MD methods solve it by creating force fields that can approximate the system's potential energy by using the positions of the particles in the system and parameters that are fit to the system.[4] The parameters for a force field are fit to the chemical system based on quantum chemical calculations and empirical data of the system. The fact that force fields need to be tuned to a chemical system before they can be used to describe it presents several limitations. First of all, it becomes difficult to create a force field that accurately describes many different kinds of interactions. This means that force fields are often limited to one kind of system and have trouble describing systems with many different types of interactions. Second, fitting the parameters to create an accurate force field requires a high level of knowledge of the system, making it difficult to use them to investigate new or unknown systems. Third, force fields contain no information on the electronic structure of the system instead they only use the positions and fitted parameters to describe the interatomic interactions.[5] This can represent a problem if you want to study a system where the electronic structure changes during a simulation, e.g. a bond is created or broken. In many force fields, the atoms are explicitly bonded together. This makes it very difficult for these force fields to describe a system before and after a change in its electronic structure occurs, as it would only be fit to describe one of the two states. There do exist reactive force fields, like ReaxFF, which can describe the change in a system's bond structure.[6–8]

Compared to classical MD, *ab initio* MD takes another approach to approximate the interatomic interactions. In *ab initio* MD the forces acting on the atoms in the system are calculated at each time step with quantum mechanical methods, like density functional theory (DFT), that uses the system's electronic structure.[3,9] This means that the electronic structure of the system is used at each time step, and *Ab initio* MD can therefore handle changes in it, unlike classical MD. However, it is practically impossible to completely solve the quantum mechanics of the system and approximations are needed to calculate the forces using quantum mechanical methods. This means that *ab initio* MD calculates forces by approximating the quantum mechanics of the system, instead of approximating the interatomic interactions using force fields like in classical MD.[3,9] This yields advantages like the fact that *ab initio* MD can be used to simulate changes in bonds and that it does not need to be fit to a system. Because of this, *ab initio* MD can be used to obtain new information as extensive knowledge of the examined system is not required. However, these advantages do come at a cost, as it is very computationally heavy to calculate the forces with quantum mechanical methods at each time step.[3] Even with the large computational requirement of *ab initio* MD the rise in computational power and parallel computing over the last decades means that the use of *ab initio* MD has become widespread in the scientific community. *Ab initio* MD is being used in drug discovery, biomolecular modeling, and material science.[10–13]

The limitations of the computational requirements of *ab initio* MD are most notable in its timescale. There is a limit to how large a time step you can use, as using too large a time step results in too much inaccuracy and an inability to describe the short-term vibrations of the system. With a ceiling for the length of time steps in MD, the timescale of a simulation is dependent on the computation time for each time step in the simulation. The computation time for a time step is dependent on the number of particles in the MD simulation, the computational power available, and whether classical or *ab initio* MD is performed. In general, it is possible for *ab initio* MD simulations to achieve a timescale in the order of picoseconds while classical MD can achieve timescales of nanoseconds even up to a few microseconds.[13,14] The limited timescale of *ab initio*MD means that it is difficult to probe certain phenomena that happen at a microsecond scale. This means that certain phenomena, that would otherwise be primed for the use of MD to explore occur over too large a timeframe for *ab initio* MD to easily be used. There exist various methods, like bond-boost, hyperdynamics, and temperature-accelerated dynamics, which can be used to increase the timescale of MD. While these methods do expand the timescale of *ab initio* MD they come at a cost in the accuracy of the dynamics from time step to time step.[14–16]

Another problem that comes from the short timescale of MD is how to describe rare events. An example of this is crystal nucleation, an important part of studying the mechanics behind material growth. For a simulation with 100 000 particles in a $12 \times 12 \times 12\,\mathrm{nm}^3$ box the nucleation rate would be about $10^{-39}\,\mathrm{fs}^{-1}$.[13] This is much too small for crystal nucleation to occur with any regularity in *ab initio MD*, or even classical MD, simulations. Just like with the limited timescale, there exist methods to improve the sampling of rare events. This could be by directing the simulation on a pathway toward a specific end product or by taking a trajectory that undergoes nucleation and samples many small perturbations of it. Both of these methods have major drawbacks, especially because the MD is not allowed to freely explore the state space of the system but is instead directed or limited to a small area of the state space.[13]

It is also possible to increase the time frame of MD simulations by doing classical MD instead of *ab initio* MD. However, classical MD comes with the disadvantages mentioned earlier. There has been a lot of work and promise to try and alleviate some of these problems by using machine learning (ML) to create the force fields. While the ML force fields do show promise they are still very specific to a system and require a large amount of data describing the system as well as training time.[17,18] With all this talk about things *ab initio* MD cannot do with its current computational limits, it is also important to remember that speeding up the computation is also beneficial to the work already applying *ab initio* MD. Even if a speed-up does not open up a new time domain a reduction in computational requirements will lead to an ease of entry into the field, allowing even more non-specialized groups to use *ab initio* MD, and a reduction in time needed to get results boosting the productivity of the field as a whole.

In this thesis, I worked on creating a model that could use auto-regression (AR) on the force trajectories of an *ab initio* MD simulation to predict the forces at future time steps. This will allow the simulation to skip using DFT to find the forces on the system for a certain number of time steps each time the model is applied. A similar model was created by Brutovsky *et al.*[19,20] where they used auto-regression to predict the non-bonding forces in *ab initio* MD. However, in their model, the bonding forces would still have to be explicitly calculated at each time step limiting the speed-up gained from applying the model. My model will predict the total force acting on the atoms in

the simulation. This means that the only other calculations that need to be performed on the time steps the model is applied are solving Newton's equations of motion. An AR model is a way to predict the new values in a time series based on its last p values.[21] The goal of the model is to have it be generally applicable to many different chemical systems without needing any prior fitting to the specific system before it is applied To achieve this a single instance of the model is not expected to be able to describe every variation of a system. Instead, each instance of the model describes a small window of the force trajectory by fitting it to the previous i points of the trajectory. This means that the only requirement for the model to be applied to any system is that the *ab initio* MD simulation has been run for i time steps. Because the model will in practice replace explicitly calculated time steps the speed-up gained by applying the model will be limited by the number of time steps that it can replace in a simulation. This is limited by how often the model can be applied, i.e. the number of time steps that are needed to fit the model, and the number of time steps that the model can accurately predict. This will likely result in a smaller speed-up than if methods that have been trained on the system before the simulation, like force fields, are used. The goal is therefore to create a model that provides a smaller speed-up than more labor-intensive methods, like force fields, but can be applied to all systems without any set-up work required.

As chemical systems are very chaotic it can be difficult to evaluate the performance of a model. A big part of this thesis has therefore been focused on how to evaluate whether a model is accurate enough that the speed-up it gives outweighs the new errors introduced. The chemical system primarily used in this thesis is ethanol in the gas phase. This system was chosen in part because of its small size, ensuring quick calculations that make it easier to gather data when testing the model, but also because it still has a varying energy landscape. Additionally, ethanol has been used by Chmiela *et al.* as a test system for their ML force field model allowing for comparison between the performance of my and their model.[22]

2 Predicting Molecular Forces with a Model

A time series is a sequence of data equally spaced in time. Trajectories from MD fit this criterion as they contain data describing the chemical system at points in time separated by the chosen time step. Times series are present everywhere and it is always important to predict the future. Because of this, forecasting time series is an important part of modern society, with it being used to predict the weather, stocks, energy consumption, and in many different scientific disciplines.[21,23–27] In time series forecasting a model is created that given a time series $X = (x_0, x_1, x_2, \ldots, x_{t-1})$ can predict the future values of the series X; $x_t, x_{t+1}, \ldots, x_{t+N}$. When forecasting a time series the model used should be chosen based on the specific time series to get the most accurate result. For some time series, it is possible to create a model based on physical laws that can predict all future points in the time series exactly. For example, if you have a time series of the positions of a thrown projectile you can use the equations of motion to calculate all future positions of the projectile. While this is nice in theory, in practice, things become more complicated and it is close to impossible to deterministically predict all future values of any time series. If we take the projectile example again, you can deterministically determine the effect of gravity but you can only approximate the wind resistance as sudden gusts of wind can change the force applied by the wind to the projectile.[21] Depending on the complexity of

the time series predicting the future values using a model based on physical laws might be accurate enough. However, if the time series is very complex it might be better to use a model with parameters that are directly fit to the data. It should be noted that the more complex and chaotic a system the time series is describing the harder it is for a model to predict long into the future as it gives more opportunity for something unexpected, e.g. a sudden gust of wind, to happen. Given that chemical systems are so complex it does not make sense to create a model based on the physical laws, as that is basically what is already done by DFT and other simulation software used in *ab initio* MD. If you want to create a model that can forecast an MD trajectory quicker than the software, it would be prudent to create a model with parameters that have been fit to the previous values in the trajectory.[21]

2.1 Auto-Regressive models

A common way to fit a model to past data in a time series is by using an AR model. An AR model is a regression model where the predicted value is regressed on the previous values in the time series. An example of this is predicting the number of products sold in a month based on the sales number of the previous months. The general expression for an AR model is

$$AR(p) : x_t = \sum_{i=1}^{p} (\theta_i x_{t-i}) + a, \tag{1}$$

where p is the order of the model, which denotes the number of points in the past the AR model uses to predict the next point in the time series, θ are fitted parameters, x are values in the time series, and a is a constant. To get a well-performing AR model it is important to choose an order, p, that fits the specific time series the model is forecasting. If a too-low order is chosen you miss information that can help the model. On the other hand, if the order chosen is too large the model will become overly complicated and contain many parameters which does not have any useful information. In general, it is important to use as few parameters as possible to describe the trend in a time series. If a model has more parameters than are needed, it can overfit to the training data making it ill-applied to predict values it is not trained on.[28]

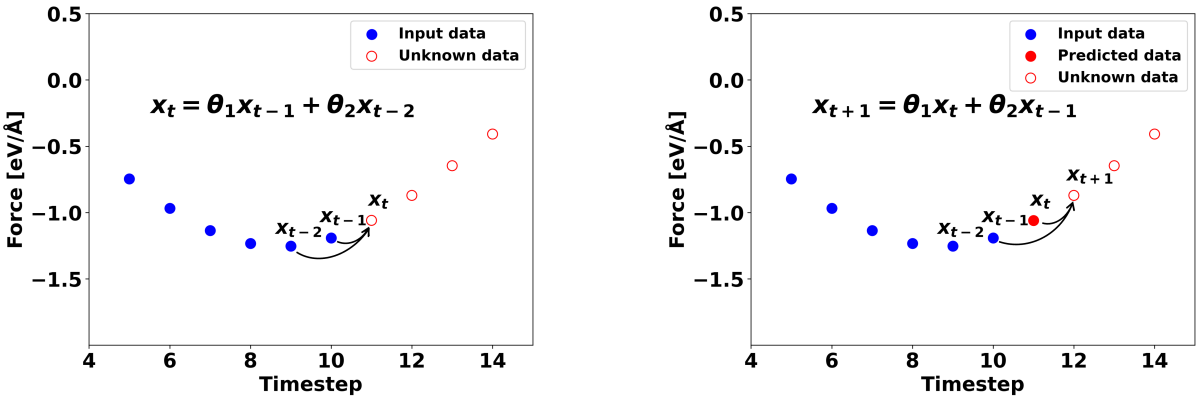
To give an example of how an AR model looks, an AR model with the order $p = 2$ takes the form

$$AR(2) : x_t = \theta_1 x_{t-1} + \theta_2 x_{t-2} + a. \tag{2}$$

Figure 1 shows how an AR(2) model is used to predict points in a time series. Here it can be seen that an AR model can be used to predict infinite future points given enough starting data. When predicting further than one point in the future the AR model will use points it has previously predicted as input to predict the next points. For example, when predicting the second point in the future an AR(2) model will look like this

$$AR(2) : x_{t+1} = \theta_1 x_t + \theta_2 x_{t-1} + a. \tag{3}$$

The fitted parameters θ are the same as was used when predicting the first point, the only thing that has changed is that the points used as input have been shifted such that the data point from the original time series x_{t-1} and the predicted point x_t is used to predict x_{t+1}.[28]



(a) (b)

Figure 1: The blue dots are data in a force trajectory while the red dots are values predicted by an AR(2) model. In a) only data originally in the time series is used to predict the next point. In b) a point previously predicted by the model is used to predict the next point.

2.1.1 Vector Auto-Regression

One problem with AR models is that they only incorporate information from the time series they are forecasting. In many cases, there are other time series that contain information that allows for better predictions by the AR model. For example, when creating a model to forecast the weather, temperature, pressure, and precipitation are all interconnected and contain information that can help predict each other. Luckily it is fairly simple to expand an AR model to include information from other time series. If we again take the example of an AR(2) model forecasting the time series $X_1 = (x_{0,1}, x_{1,1}, \ldots, x_{t-1,1})$ we can expand it to include information from another time series $X_2 = (x_{0,2}, x_{1,2}, \ldots, x_{t-1,2})$ by doing the following

$$x_{t,1} = \theta_{1,X_1} x_{t-1,1} + \theta_{2,X_1} x_{t-1,2} + \theta_{3,X_1} x_{t-2,1} + \theta_{4,X_1} x_{t-2,2} + a_{X_1}. \tag{4}$$

However, using this new model we run into a problem if we want to be able to predict more than one future value. The model does predict $x_{t,1}$ but it does not predict $x_{t,2}$. This means the model is missing a value when trying to predict $x_{t+1,1}$. Because of this if we want to be able to predict more than one point in X_1 we also need a model that can forecast X_2

$$x_{t,2} = \theta_{1,X_2} x_{t-1,1} + \theta_{2,X_2} x_{t-1,2} + \theta_{3,X_2} x_{t-2,1} + \theta_{4,X_2} x_{t-2,2} + a_{X_2}. \tag{5}$$

We have now created two multivariate models that can use information from each other's time series to forecast any number of points in the future. In this example, the multivariate models only used two time series, but it can be extended to any number of time series. Instead of writing each model on its own, we can combine them using vector notation which creates a type of model called vector auto-regression (VAR). If we write a VAR model with an order of $p = 1$ and generalize it to contain any number of time series, it would look like this:

$$VAR(1): \begin{bmatrix} x_{t,1} \\ x_{t,2} \\ \vdots \\ x_{t,N} \end{bmatrix} = \begin{bmatrix} \theta_{1,X_1} & \theta_{2,X_1} & \ldots & \theta_{N,X_1} \\ \theta_{1,X_2} & \theta_{2,X_2} & \ldots & \theta_{N,X_2} \\ \vdots & \vdots & & \vdots \\ \theta_{1,X_N} & \theta_{2,X_1 N} & \ldots & \theta_{N,X_N} \end{bmatrix} \begin{bmatrix} x_{t-1,1} \\ x_{t-1,2} \\ \vdots \\ x_{t-1,N} \end{bmatrix} + \begin{bmatrix} a_{X_1} \\ a_{X_2} \\ \vdots \\ a_{X_N} \end{bmatrix}. \tag{6}$$

We can also write a general expression for VAR models with any order p

$$VAR(p): \begin{bmatrix} x_{t,1} \\ x_{t,2} \\ \vdots \\ x_{t,N} \end{bmatrix} = \sum_{i=1}^{p} \left(\Theta_i \begin{bmatrix} x_{t-i,1} \\ x_{t-i,2} \\ \vdots \\ x_{t-i,N} \end{bmatrix} \right) + \begin{bmatrix} a_{X_1} \\ a_{X_2} \\ \vdots \\ a_{X_N} \end{bmatrix} \tag{7}$$

where Θ are $N \times N$ matrices containing the fitted parameters θ. In order to differentiate between general VAR models and the VAR model I created to forecast molecular forces, I will call my model VAR_mol.[29]

2.2 Creating the Model

If we want to be able to skip using quantum mechanical methods to calculate the forces for a time step in an *ab initio* MD simulation, we need to predict the forces on all of the atoms in the system. As all of the atoms are also part of the same interconnected system the different force trajectories in the system are likely correlated. Because of this, for my thesis, I will be using my VAR_mol(p) model to forecast the force trajectories. This will allow me to forecast the forces of all atoms using only a single model, instead of having to create a separate model for each force trajectory that needs to be predicted. It will also allow the model to forecast a force trajectory using information from every force trajectory in the system.

In order to optimize the hyperparameters of the VAR_mol model and to evaluate its performance, trajectories from *ab initio* MD simulations are required. As mentioned in the introduction the primary chemical system I worked with in this thesis is ethanol in the gas phase. The *ab initio* MD simulations were done using ASE[30] to solve Newton's equations of motion, as well as running and storing the trajectories. GPAW[31] was used to calculate the forces at each time step using DFT. Before any *ab initio* MD simulations were done, the ethanol molecule was geometrically optimized using the BFGS optimizer from ASE. The convergence criterion was set with the keyword $fmax = 0.01$, meaning that all forces in the system must be below $0.01\,\frac{\text{eV}}{\text{Å}}$ for the geometry optimization to converge. The calculator used for the geometry optimization was from GPAW. PBE was used as the exchange correlational function and double zeta polarized (dzp) as the basis set. Basis sets are sets of functions that are used to describe the electronic wave function by DFT. The k-points used in the calculator were $(1, 1, 1)$, and the mode keyword was set to LCAO. The occupations keyword was set to FermiDirac(0.0) and for the mixer keyword, Mixer(0.02, 5, 100) was used, both are functions from GPAW. The convergence was set to $1 \cdot 10^{-8}$ for both density and energy. The unit cell was set to have periodic boundary conditions and the molecule was centered with 10 Å of vacuum to all sides of the molecule. After the geometry optimization was completed the next step was running the *ab initio* MD simulations. Before each simulation, the momenta of the atoms were randomized using the Maxwell Boltzmann distribution at 400 K. The *ab initio* MD simulations were done using the Velocity Verlet (VelVer) scheme from ASE with a time step of 0.5 fs. The calculator used in the simulation was the same as the one used in optimization except for the mixer keyword not being set, the convergence being $1 \cdot 10^{-7}$ for both energy and density, the occupation keyword was FermiDirac(0.05), and a single zeta (sz) basis set being used which is less accurate than dzp but also computationally cheaper.

The fact that the geometry optimization and *ab initio* MD were done with different basis sets could cause some problems. The structure gained after geometry optimization with dzp will have a force of less than $0.01\,\frac{\text{eV}}{\text{Å}}$. However, as dzp and sz are different basis sets, it is not certain that the structure from a geometry optimization using dzp will also have forces less than $0.01\,\frac{\text{eV}}{\text{Å}}$ when described with sz. This can then result in the *ab initio* MD simulations starting with a force that pulls the atoms toward the local energy minimum in sz instead of the structure being at equilibrium. When I made the geometry optimization and the *ab initio* MD simulations I did not realize that a structure at rest in dzp will not necessarily be at rest in sz. I wanted to use dzp in the geometry optimization to make sure that it was as accurate as possible. For the *ab initio* MD simulations I wanted to use sz to reduce the computation time for each time step making it easier to gather data for testing the model, which was also why a small molecule like ethanol was chosen as the primary test system. However, the fact that dzp was used in the geometry optimization instead of sz should not have a big impact on the overall results. All *ab initio* MD simulations done used the same optimized geometry, meaning that any force induced by the structure not being at equilibrium is the same for all of the *ab initio* MD simulations. In addition to this the *ab initio* MD simulations are not calculating a chemical property, but instead simulating how the atoms move over time. The small force induced by the molecule not being in an energy minimum will probably not have a great effect on the overall dynamics of the system, especially considering that the initial momenta of the atoms are set using Maxwell-Boltzmann distribution and the *ab initio* MD simulations are run for more than 10000 time steps. Overall the *ab initio* MD simulations should still be valid but in retrospect, it would have been better to run both the geometry optimization and the *ab initio* MD simulations with the same basis set. However, when I realized this there was not enough time to run the geometry optimization and *ab initio* MD simulations again.

2.2.1 Fitting the Model's Parameters

Having decided the type of model to be used the next step is finding a way to fit the parameters Θ to the data. A good and simple way of fitting parameters is by using the least square method. Here the parameters are chosen such that they minimize the sum of the square difference between data values, y, and the values predicted by the VAR_mol(p) model, $\tilde{x}$

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{t-p} (\tilde{x}_{t-i} - y_{t-i})^2 \tag{8}$$

where t is the number of time steps used to fit the model and p is the order of the model. When working with linear models it is helpful to write it as a system of equations in matrix form.

$$A\Theta \approx y \tag{9}$$

where A is a matrix containing the input data the model needs to predict the next value, Θ is a matrix containing the parameters of the model, and y is a matrix consisting of the data values that the model tries to predict. The linear least square method can be used to find the parameters, Θ, that minimize

$$\underset{\Theta}{\text{minimize}} \, \|A\Theta - y\|^2. \tag{10}$$

One way of finding the parameters Θ that fulfill the least squares condition is by using the normal equation

$$A^T A\Theta = A^T y. \tag{11}$$

The parameters that are optimal for the least square method can then be found by using $\left(A^T A\right)^{-1}$

$$\Theta = \left(A^T A\right)^{-1} A^T y. \tag{12}$$

Because my VAR_mol model is linear and all the time series in the model are predicted using the same input parameters we can write it in the aforementioned matrix form. As an example, a VAR_mol(2) model consisting of 2 time series and using data from $t = 5$ time steps as input looks like this

$$A\Theta = y \Rightarrow \begin{bmatrix} x_{2,1} & x_{2,2} & x_{1,1} & x_{1,2} \\ x_{3,1} & x_{3,2} & x_{2,1} & x_{2,2} \\ x_{4,1} & x_{4,2} & x_{3,1} & x_{3,2} \end{bmatrix} \begin{bmatrix} \theta_{1,X_1} & \theta_{1,X_2} \\ \theta_{2,X_1} & \theta_{2,X_2} \\ \theta_{3,X_1} & \theta_{3,X_2} \\ \theta_{4,X_1} & \theta_{4,X_2} \end{bmatrix} = \begin{bmatrix} x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \\ x_{5,1} & x_{5,2} \end{bmatrix}. \tag{13}$$

This example just showed one specific VAR model, but it can be generalized to describe VAR_mol models with any order, number of time series, and inputs. In the general form, A takes the shape $N_{eq} \times N_{obs}$, Θ takes the shape $N_{obs} \times N_{traj}$, and y takes the shape $N_{eq} \times N_{traj}$. Where $N_{eq} = t - p$ is the number of equations used to fit the model, $N_{obs} = N_{traj} \cdot p$ is the number of observables in the model, and $N_{traj} = N_{atoms} \cdot 3$ is the number of force trajectories in the system. If we take the case of creating a VAR_mol(2) model for ethanol with $t = 12$ it results in $N_{traj} = 27$, $N_{obs} = 54$, and $N_{eq} = 10$. This is an underdetermined system of equations meaning that there are infinite combinations of parameters that fulfill $\|A\Theta - y\|^2 = 0$. In this case, we need another way of finding a single solution. One way to do this is to choose a solution that minimizes the norm of the parameters in the model

$$\underset{\Theta}{\text{minimize}} \ \|\Theta\|^2 \tag{14}$$

while still fulfilling

$$\|A\Theta - y\|^2 = 0. \tag{15}$$

However, this creates a new problem, as applying the normal equation does not give this solution. Luckily this can be solved by using singular value decomposition (SVD). SVD is a way of describing a matrix using 3 other matrices

$$A = U\Sigma V^T \tag{16}$$

where Σ is a diagonal matrix consisting of the singular values σ_i, and U and V are both unitary matrices. A matrix being unitary means that the inverse of said matrix is equal to the complex conjugate

$$U^* U = V^* V = I. \tag{17}$$

In the case of the matrix M being real, which it is in the case of it consisting of data from force trajectories, the complex conjugate of U and V is equal to the transpose

$$U^T U = V^T V = I. \tag{18}$$

To see how SVD solves the least squares problem we can replace A in the normal equation with the SVD of the matrix

$$V\Sigma^T U^T U\Sigma V^T \Theta = V\Sigma^T U^T y. \tag{19}$$

Using the fact that U and V are unitary matrices, and utilizing the fact that A is a real matrix in my case, we can reduce the expression

$$V\Sigma^T\Sigma V^T\Theta = V\Sigma^T U^T y \Leftrightarrow \Sigma^T\Sigma V^T\Theta = \Sigma^T U^T y. \tag{20}$$

Based on this an expression for finding the parameters Θ that minimizes the square of the errors and norm of the parameters can be created

$$\Theta = V\left(\Sigma^T\Sigma\right)^{-1}\Sigma^T U^T y. \tag{21}$$

A big advantage of using SVD to find the parameters is that it finds the solution that fulfills the least square condition if the system is overdetermined. However, if the system is underdetermined it will find the parameters that fulfill the minimal norm condition. My model therefore uses SVD to fit the parameters Θ to the data as it is better at handling underdetermined systems than the normal equation is.[32,33]

2.2.2 Regularization

As we are working with an underdetermined system it is easy for the model to be overfit to the data. Overfitting a model can hurt its ability to predict data outside of the training data. Regularization is a category of tools used to combat the overfitting of models with many parameters. There are different ways of implementing regularization depending on the model and the effect desired. One of the most used ways is Tikhonov regularization, also called L2 or ridge regression.[34,35] Because it is the method I implemented it will be the one I focus on in this section. Tikhonov Regularization works by adding a term to the error function that punishes the model for having too large parameters

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{t-p}(\tilde{x}_{t-i} - y_{t-i})^2 + \sum_{i=1}^{n}\lambda|\theta_i|^2 \tag{22}$$

where n is the number of parameters and λ is the regularization factor. The strength of the regularization can be tuned by changing λ. As Tikhonov regularization punishes large parameters it will result in the predictions by the model being closer to 0 than if it was not applied to the model. The extent to which the predictions will approach 0 is dependent on the strength, λ, of the regularization used. The force trajectories of *ab initio* MD simulations oscillate back and forth with a mean of 0. It will therefore likely be beneficial to the model that Tikhonov regularization forces the predicted forces to approach 0.[34,35]

Tikhonov regularization can easily be implemented using SVD just by adding an identity matrix multiplied by λ

$$\Theta = V\left(\Sigma^T\Sigma + \lambda I\right)^{-1}\Sigma^T U^T y. \tag{23}$$

In the case of $\lambda \to 0$, the parameters will be the same as the ones achieved from least norm optimization. Because of this, it is easy for the same code to use SVD for optimizing the parameters of a model with and without Tikhonov regularization, the only thing that needs to change is the value of λ used.[34,35]

The code used to fit the parameters can be seen in Appendix A. The overall structure of the code that creates a class for the model was received by Ph.D. student William Bro-Jørgensen but the part of the code that uses SVD to fit the parameters was created by myself.

2.2.3 Integrating Model with ASE

Before using the model to speed up an MD simulation it must be integrated with the simulation scheme. As written earlier, the scheme used to create the simulations in this thesis is the VelVer algorithm from ASE. The VelVer algorithm is an integration algorithm that solves Newton's equations of motion and calculates the positions and velocity of the atoms at all time steps

$$r(t + \Delta t) = r(t) + v\left(t + \frac{1}{2}\Delta t\right)\Delta t, \tag{24}$$

$$v(t + \Delta t) = v(t) + \frac{F(t) + F(t + \Delta t)}{2m}\Delta t \tag{25}$$

where r denotes the positions of atoms, v is the velocity of the atoms, F is the force acting on the atoms, m is the mass of the atoms, and Δt is the time step. When the VelVer algorithm calculates the positions and velocities at each time step in the simulation it does so by iterating over the following steps:

$$\text{Step 1: } v\left(t + \frac{1}{2}\Delta t\right) = v(t) + \frac{F(t)\Delta t}{2m}. \tag{26}$$

$$\text{Step 2: } r(t + \Delta t) = r(t) + v\left(t + \frac{1}{2}\Delta t\right)\Delta t. \tag{27}$$

$$\text{Step 3: Evaluate } F(t + \Delta t). \tag{28}$$

$$\text{Step 4: } v\left(t + \Delta t\right) = v\left(t + \frac{1}{2}\Delta t\right) + \frac{F(t + \Delta t)\Delta t}{2m}. \tag{29}$$

These steps are repeated until the desired number of time steps have been evaluated. When the VelVer algorithm calculates the positions of atoms at the next time step it uses the average velocity between the time steps instead of the velocity at the initial or final time step.[36]

When applying the VelVer algorithm in an *ab initio* MD simulation, step 3 will be done using DFT or similar methods. It is at this step the simulation will be sped up by using the model to predict the forces instead of calculating them with DFT. However, in order to apply the model and speed up the *ab initio* MD simulation the VelVer algorithm will have to first use DFT to evaluate the forces. This must be done for enough time steps so that enough data to fit the model has been gathered. The model can then be used to predict the forces for the next time step. The exact number of time steps needed to be evaluated by DFT depends on the number of inputs, t, needed to fit the model. If a model has $t = 10$ DFT needs to evaluate 10 time steps each time the model is applied to ensure that good data is used when fitting the model. The code that integrates the model with ASE's VelVer algorithm can be seen in Appendix B.

2.2.4 Optimizing Hyperparameters

Now that all preliminary work for using the model in *ab initio* MD simulations is done, the last part is deciding on the hyperparameters for the model. To find the best hyperparameters a grid search was performed on an *ab initio* MD trajectory of ethanol consisting of 2000 time steps. The values of parameters evaluated by the grid search were the integers in the range $[6, 15]$ for the input t, and in the range $[1, 10]$ for the order p.

For the regularization constant, λ, 30 evenly spaced values between 0 and 0.0005 were evaluated. The reason for the values of λ being so low is that preliminary investigations showed that any stronger regularization had a clear negative effect on the model. Each model's performance in predicting the forces for up to 8 time steps in the future was evaluated. Based on the grid search the hyperparameters for the model with the lowest mean absolute error (MAE) in predicting the forces for the first 6 time steps were $t = 15$, $p = 5$, and $\lambda = 8.572 \cdot 10^{-5}$.

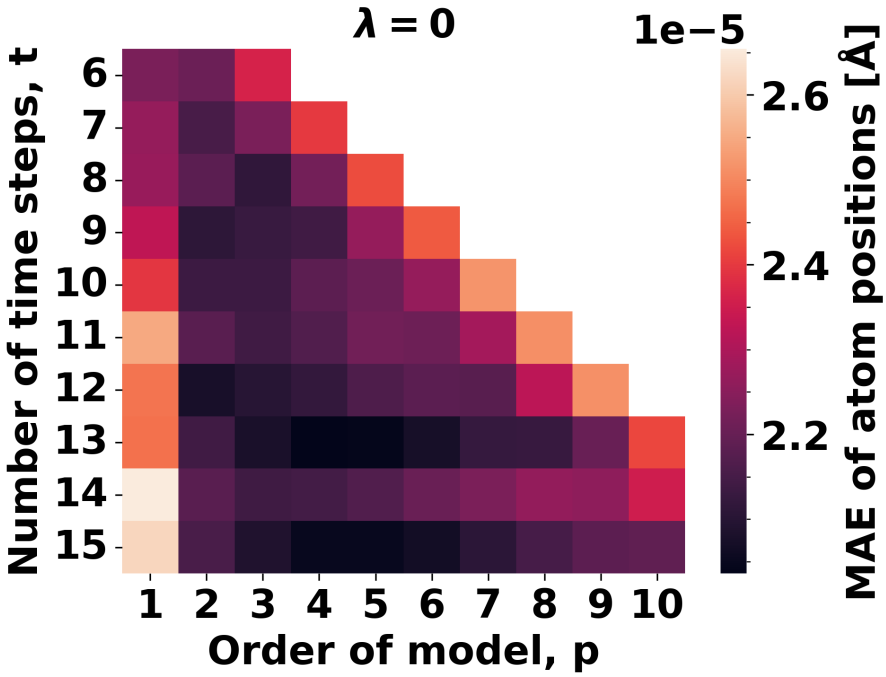


Figure 2: The grid search found a couple of areas with lower MAE than their surroundings. The heatmap shows the MAE of models with different values for input and order, and with $\lambda = 0$, predicting 1 time step. The MAE shown in the heatmap is the difference in the positions of the atoms between the values gained from simulation and from predicting the forces with the model. The color scale has been rescaled to highlight the differences between models better.

In order to get a better understanding of how the performance of the model changes with input and order a heatmap with the MAE of models with $\lambda = 0$ was created. The heatmap can be seen in Figure 2. The regularization constant, λ, was set to 0 because even though the optimal value for λ varied from model to model, it was always very low. The models in the heatmap only predict a single time step. This is done because if the first step predicted is bad it will propagate to the later time steps. It is therefore paramount for the model to predict the first step accurately. In addition to this what is important for the success of the model is the deviation in the positions of atoms after the model is applied, and not the deviation in the forces predicted. This is because the positions of atoms, rather than their momenta and forces at previous time steps, are what will be used by DFT to calculate the forces in later time steps.[9] Because of this the MAE shown in the heatmap is calculated from the deviation in the positions of the atoms dependent on whether the model or DFT was used to evaluate the forces for the time step. This will only be the case for this figure and for the rest of the thesis MAE will be based on the deviation of the forces predicted by the model and evaluated by

Table 1: Hyperparameters of models

	Input (t)	Order (p)	λ
Model 1	12	2	0
Model 2	12	2	$1.16 \cdot 10^{-4}$
Model 3	15	4	0
Model 4	15	4	$8.572 \cdot 10^{-5}$

DFT, as that is easier to calculate. In the heatmap in Figure 2 it can be seen that there is a low point in MAE around $t = 15$, and $p = 5$ like we would expect as that were the best-performing hyperparameters in the grid search. There is a similar low point around $t = 13$, $p = 5$. In addition to this $t = 12$, $p = 2$ also performs well while having a smaller input than the other models.

Models with two different sets of input and order were chosen to be tested in *ab initio* MD simulations based on the grid search. For each set, a regularized version, with the optimal λ for that input and order, and a non-regularized version were tested. The first set chosen was $t = 15$, $p = 5$, as the regularized version of this set of hyperparameters performed the best in the grid search. The other input and order sets were $t = 12$ and $p = 2$, as they performed well and had a smaller input parameter than the other well-performing models. It is beneficial to have a model with a small input, as the smaller the input of the model the more often it can be applied and therefore achieve a greater speed-up. The hyperparameters of the models tested in simulations can be seen in Table 1. Overall the different models performed very similarly in simulations and it was difficult to say that one was better than the others. Because of this, and to save space, the rest of this thesis will primarily focus on the $t = 12$, $p = 2$, and $\lambda = 0$ model as it has the joint smallest input and it seemed to perform slightly better than the other tested models.

2.3 Error Evaluation

Having now decided on the hyperparameters of the VAR_mol model used it is time to evaluate the error of its predictions. The performance of the model highly depends on the input data as can be seen in Figure 3. Figure 3a shows an example where the model works well where it can be seen that the predictions the model made roughly follow the forces calculated by DFT. In contrast, Figure 3b shows an example where the model breaks down. The first two predictions made by the model are close to the DFT calculated forces but after this, the predictions quickly deviate from the correct values. Based on this and other snapshots of the model, it seems the model breaks down when the input data ends just before the maximum or minimum of a curve. When this happens the model expects the amplitude of the force trajectory to keep increasing and the model overshoots the actual curve of the force trajectory. In the cases where the input data contains more information on the top or bottom of a curve, the model is better at estimating the future force trajectory.

If it was possible to estimate if the input data contained information on whether the curve had reached a maximum or minimum, it would be possible to assign different amounts of confidence to the models. If you then had high confidence in the accuracy of a model, you could use it to predict the forces of multiple time steps. If instead, you had low confidence in a model you could only predict a single step, or even not make

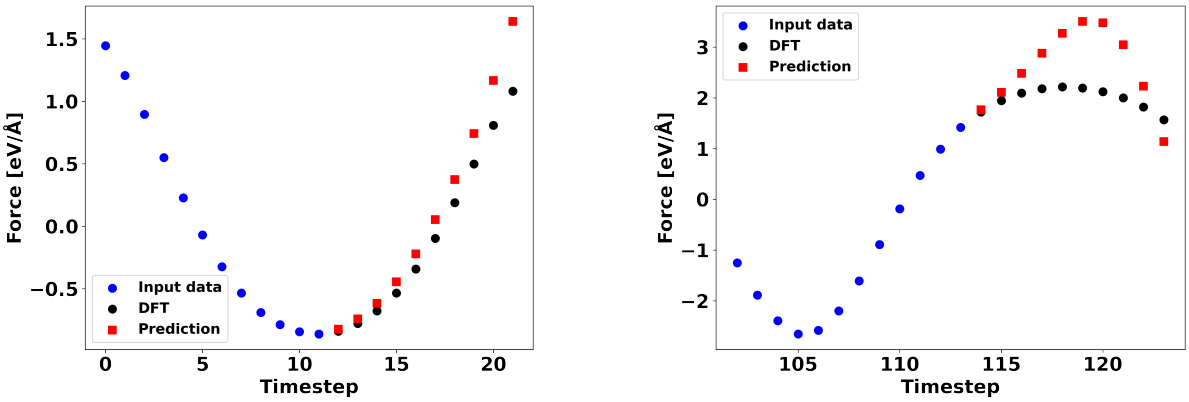
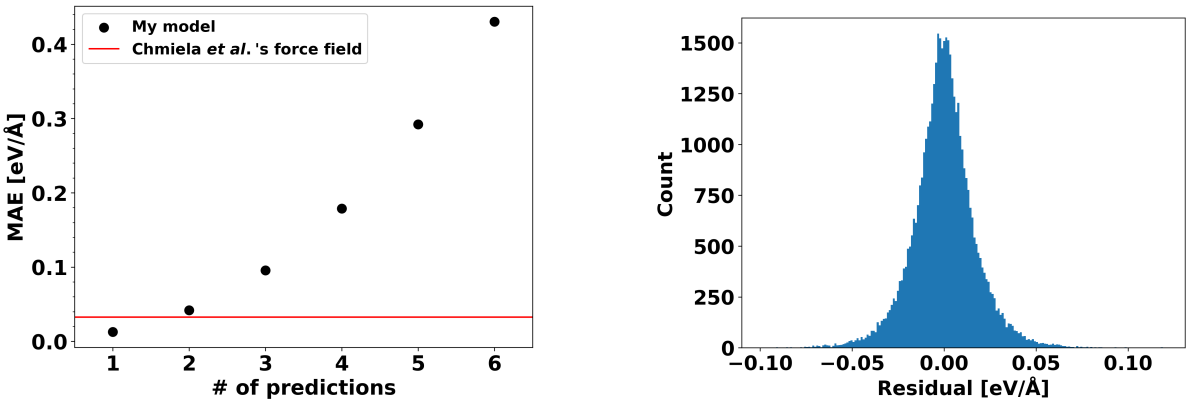


Figure 3: The VAR_mol model performs well when the input data contains information on the top or bottom of a curve. If the input data does not include information on these valleys or peaks the model performs far worse. a) shows an example of when the model performs well. b) shows an example of when the model breaks down.

any predictions and create a new model a couple of time steps later with a better set of input data. While this seems good in theory in reality it would be very difficult to predict when the curves will turn as it occurs at wildly different amplitudes throughout the trajectory. You could maybe do it by approximating the differential of the curve and increasing confidence when it gets close to 0. However, even in the case where the model performs well the error increases rapidly with the number of time steps predicted. Because of this, it is unlikely that a model with high confidence can be used to predict many more points than a low-confidence model. Any benefit gained in establishing a confidence level of the models will be small and likely outweighed by the inaccuracy of assigning the confidence level. It should also be noted that even when the model breaks down the first predictions are still good, it is only later that the model rapidly deviates from the force trajectory. Because of this, it seems that the best approach will be to always limit the number of time steps predicted by the model to one or two, no matter how good or bad the input data is.

This trend can also be seen if we examine the VAR_mol model quantitatively instead of qualitatively. Figure 4a shows the MAE for the model's predictions at different time steps. Here it can be seen that the MAE exponentially increases the further in the future the model predicts. The horizontal red line in Figure 4a represents the ML force field created by Chmiela *et al.* which has an MAE of approximately $0.0172\,\mathrm{eV}\cdot\mathrm{Å}^{-1}$.[22] My model has a lower MAE of approximately $0.012\,\mathrm{eV}\cdot\mathrm{Å}^{-1}$ when predicting the first time step, while it has a higher MAE of approximately $0.042\,\mathrm{eV}\cdot\mathrm{Å}^{-1}$ when predicting the second time step. For any points further in the future, my model performs drastically worse than the force field by Chmiela *et al.* When comparing the MAE of my model and the force field by Chmiela *et al.* it is important to remember that they are fundamentally two different types of models. The force field by Chmiela *et al.* is applied in classical MD and is therefore used to find the forces in all time steps of the simulation. My model on the other hand will only replace DFT in a fraction of time steps in an *ab initio* MD simulation. This means that the force field from Chmiela *et al.* has a higher threshold for what MAE is tolerable than my model as it results in a bigger speed-up. However, my model also has some advantages. First of all, is the fact that my model is



(a) (b)

Figure 4: The VAR_mol model's MAE is lower than the MAE of the ML force field by Chmiela *et al.* when it only forecasts a single time step. The residuals of the VAR_mol model are nicely centered around 0 meaning it has no statistically unexpected errors. Figure a shows the model's MAE for different numbers of predictions made. The red line corresponds to the MAE achieved by Chmiela *et al.*'s force field.[22] Figure b shows the distribution of the input = 12, order = 2 model's residuals.

generally applicable to all chemical systems and does not require any training before it is implemented in a simulation. It also keeps the electronic structure intact and therefore retains the advantages *ab initio* MD has over classical MD.

Based on how quickly the MAE increases with the number of steps predicted it seems prudent to only predict the forces of a single time step each time the model is applied. The fact that the MAE of predicting one time step with the model is lower than the MAE of the ML force field Chmiela *et al.* is also reassuring. Only predicting a single time step will hopefully also help prevent bad input data, like in Figure 3b, from having a big impact on the overall performance of the model. To check whether the model breaks more often than is expected a histogram of the residuals of the model predicting one time step is shown in Figure 4b. Here it can be seen that the residuals of the model are nicely centered around 0 and there are no unexpected peaks at either end that could be from the model breaking down. Based on this the model does not break down more often than what would statistically be expected and it is ready to be applied in actual *ab initio* MD simulations.

2.4 Evaluating Speed-up of Model

As mentioned in the last section it is important to know the speed-up gained by applying a VAR_mol model in order to evaluate whether the errors it introduces are acceptable. The speed-up gained by the model can be found by using the average time it takes to do a time step in an *ab initio* MD simulation with and without the model

$$S = \frac{t_{DFT}}{t_{combined}} \tag{30}$$

where $t_{combined}$ and t_{DFT} denote the time it takes to calculate an average time step in a simulation with and without the model. The time it takes to calculate an average timestep with DFT, the model, and a step in a simulation combining the two can be

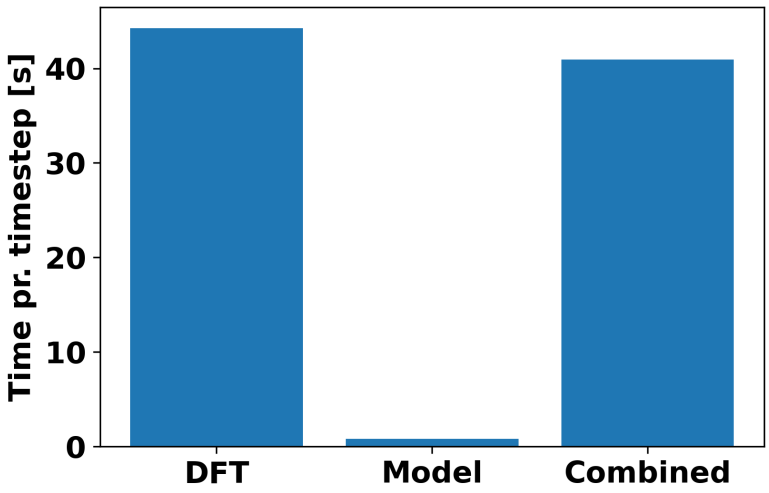


Figure 5: Bar plot showing the average time it takes to calculate a time step. The DFT bar shows the time it takes for DFT. The model bar shows the time it takes my model. The combined bar shows the average time it takes in a simulation using both DFT and my model. The time it takes for the VAR_mol model to calculate a time step is much smaller than the time it takes DFT. The time an average time step takes in a simulation incorporating the model is close to the DFT time as the model is only applied once every 12 time steps.

seen in Figure 5. Here it can be seen that a DFT time step takes much longer, 44.27 s, compared to fitting and predicting the forces using the model 0.8 s. The time it takes to solve the equations of motion is included in both times.

$$t_{DFT} \gg t_{model}. \tag{31}$$

Because of this large difference in time, the speed-up gained from the model can be approximated by saying that each predicted step by the model takes 0 s. The speed-up gained is then dependent on the amount of DFT steps replaced by the model. This is in turn dependent on the number of input time steps that are required to fit the model and the number of predictions the model makes each time it is applied

$$S \approx \frac{t + n_{pred}}{t} \tag{32}$$

where n_{pred} is the number of predictions made by the model. As the model will only predict one time step the approximate speed-up becomes

$$S \approx \frac{t+1}{t}. \tag{33}$$

For the model with $t = 12$ and $p = 2$ this results in the following expected speed-up:

$$S \approx \frac{12+1}{12} = 1.083. \tag{34}$$

In order to calculate the actual speed-up the average time it takes for a simulation incorporating both DFT and the model is needed. For the incorporation of the $t = 12$ and $p = 2$ model in a simulation the average time for a time step is 40.92 s. This can then

be used together with the average time for a DFT step to find the actual speed-up of applying the model

$$S = \frac{t_{DFT}}{t_{combined}} = 1.0816. \tag{35}$$

Here it can be seen that the approximation for the speed-up using Equation 33 is close to the actual speed-up. As expected the approximation overestimates the speed-up slightly as in reality, the model step does take time even if it is short. However, this difference is so small that Equation 33 still gives a good approximation of the speed-up. The approximation should also hold if it was used on a larger system. The increase in atoms will make it take longer to fit the parameters of the model, but it will also make the DFT step take longer. This makes the approximation a useful tool in estimating the speed-up one would gain from implementing the model on any given system.

3 Applying the Model in Molecular Dynamics

Having now created a model and analyzed its errors to show they are reasonable, the next step is to evaluate if it is good enough to use in a simulation. While the MAE can be used to see if the performance of the model is reasonable it does not definitively tell whether or not the model is good enough to be used in *ab initio* MD simulations. To evaluate that there needs to be established metrics that can tell whether or not the properties of the system remain unchanged after the model is applied. One problem with doing this is the fact that chemical systems are highly chaotic creating restraints on what metrics can be used and how simulations can be compared to each other. The chaotic nature of the systems means that any small perturbations of the positions of the atoms in the system will result in wildly different simulations. While the same initial positions of the atoms are used for all simulations this is not necessarily the case for the initial momenta of the atoms, as that is set using a random Maxwell-Boltzmann distribution. This means that the initial momenta of the atoms will differ across different simulations unless the same random seed is used. Even if the initial positions are the same differing initial momenta will create differing positions after a few time steps, which, due to the chaotic nature of the system, leads to wildly different simulations. Because of this, even metrics that consist of average values or distributions will vary significantly across simulations with different starting momenta, even if thousands of time steps are performed. This means that in order to isolate the effect the errors introduced by the model will have on a simulation, it needs to be compared to a simulation with the same random seed. The model should not be directly compared to how closely it matches the simulation with the same random seed as there is no objective limit to how similar they must be for the simulation not to break down. Instead, the requirement for the model to be accurate enough is that applying the model will result in a smaller change in the simulation than the difference between two *ab initio* MD simulations with different random seeds. This will ensure that applying the model keeps the simulation within the bounds of a valid solution.

However, this requirement does not indicate whether the speed-up gained by applying the model was worth the errors introduced. To evaluate this, a simple way of achieving the same speed-up as the model is needed. This simple way will then act as the baseline the model can be compared to. If the model is closer to the *ab initio* MD simulation than the baseline the speed-up gained by the model outweighs the errors introduced. In my thesis, I created the baseline by increasing the time step used in the VelVer algorithm

to achieve the same speed-up as the model. This is the easiest and simplest way of achieving the same speed-up as it does not require any model or tools outside of ASE. The time step that corresponds to the speed-up gained from the model was calculated using Equation 33. The time step used to create the same speed-up as a model with $t = 12$ on a simulation with $\Delta t = 0.5\,\text{fs}$ using VelVer is

$$\Delta t_{speed-up} = \frac{t+1}{t}\Delta t_{original} = \frac{13}{12} \cdot 0.5\,\text{fs} \approx 0.542\,\text{fs}. \tag{36}$$

Based on this there are two requirements the model must fulfill for it to be a valid method to speed up *ab initio* MD simulations. First of all the changes introduced in the simulation must be smaller than those of a simulation with different initial momenta. The second requirement is that the model must perform better than if you were to achieve the same speed-up by increasing the time step used in the VelVer algorithm. With all of these different types of simulations, an easy way to differentiate between them is needed. For the rest of the thesis, I will use model to denote a simulation applying my VAR_mol model to speed up the simulation. VelVer speed-up will be used to denote a simulation where the time step was increased to achieve a similar speed-up as the model. A pure simulation will mean an *ab initio* MD simulation that is not sped up by applying the model or increasing the time step.

3.1 Metrics for Evaluating Performance

Depending on the system you are investigating there are different things you need the MD to accurately describe. In some systems, it might be important to describe changes in bonds, to describe the different conformational isomers, or how two different molecules interact. However, generally, it is important to maintain the overall geometry of the system and the dynamics of the system. The chemical system used in this thesis is ethanol in the gas phase, which means that there are no expected changes to bonds and there are no other molecules for ethanol to interact with. The metrics of interest are therefore those that describe the overall geometry and dynamics of the system as well as its conformational isomers. There are also some metrics, such as bond length, angles, and isomer ratios, that have the advantage of being able to compare them to experimental data. These metrics are good for evaluating whether or not an *ab initio* MD simulation scheme is accurate. However, in this case, these metrics are not that valuable as the goal is to create a model that is better than increasing the time step at speeding up the simulation. Relatively comparing the model to VelVer speed-up is therefore more relevant than using experimental data to evaluate the accuracy of the model.

In the following sections, I will shortly explain the different metrics I used in my thesis to compare the VAR_mol model to pure simulations and VelVer speed-ups. Most of these metrics have been used in other papers to evaluate the performance of models. The metrics will be roughly ordered according to their complexity and usefulness in comparing the VAR_mol model to VelVer speed-ups, with the simplest metrics being presented first.

3.1.1 Average Values

One of the simplest ways of evaluating the simulations is by looking at the average values for the angles and bond lengths over the simulation. This metric, among others, was used by R. Han and S. Luber to evaluate the performance of their ML training method.[37]

One of the primary advantages of this metric is its simplicity and ease of use. It gives one number that can easily be compared to the same number from other simulations or experimental values. While the simplicity of the metric is its greatest strength it can also be its greatest weakness. In the process of boiling down a dynamic system to a single number, a lot of information is lost. In reality, what matters for the properties and geometry of the system is not the average value over the whole trajectory, it is the actual value at each time step. The same average value can come from different distributions meaning that even if the average value is the same across different simulations the distributions might be different. In this case, average values can give the false impression that the simulations are similar while in reality the angles and bond lengths occur in different distributions for the different simulations. In continuation of this even looking at the values of angles and bond lengths at each time step does not give accurate insight into the geometry or properties of the system. Angles and bond lengths are not alone enough to describe the geometry of the system. In order to fully describe the system dihedral angles and improper dihedral angles are also needed. In reality, it is not feasible to comprehend and get useful information out of such high-dimensional data without any external analysis tools. If one wanted to visually look at how the geometry of the system evolved, it would be better to just create a movie of the positions of the atoms over the simulation. In both cases, it is difficult to compare the dynamics of the system and impossible to make sure the chemical properties of the system are constant across simulations based on the values of angles, bond lengths, and dihedrals at each time step without any extra analysis tools. This all means that average values can give a quick and simple overview that might be able to spot obvious errors but they should not be used to make final decisions on the performance of a model.

3.1.2 Distributions

As mentioned in the previous section a better way to examine angles and bond lengths is by looking at their distributions instead of their average values. This gives a greater insight into what the angles and bond lengths are at each time step in the simulations. Distributions do not contain information on the sequence at which the angles and bond lengths occur in a simulation but this is not a big loss as the sequence can easily vary due to the chaotic nature of the system. However, if the energy landscape is the same throughout different simulations we would expect that the distributions of each angle and bond length will also be the same. Distributions of angles and bond lengths therefore give a much better insight into the geometry of the system than average values. The downside of this is that it requires more work to evaluate. Generally, each angle and bond length one wishes to examine the distribution of must be done so manually and often by eye. When evaluating a distribution, a plot with the histograms of each simulation of interest must be made and analyzed. This must be done for each distribution of interest resulting in a lot of manual work.

However, there do exist methods that can help narrow down the distributions of interest or make it easier to analyze the distributions. A two-sample Kolmogorov-Smirnov (KS) test can be used to compare the similarity of two distributions. The KS test will give a p-value describing how likely it is that the two samples have the same underlying distribution. This can be used to create a simple overview of how similar the distributions of the model and VelVer speed-up simulations are to the distributions from pure simulations. Based on this overview certain distributions of angles and bond lengths of interest

can be chosen for further analysis. One problem with the two-sample KS test is that the p-value given by it will generally become smaller when the size of the samples becomes larger. This happens because the KS test becomes more certain with more data and any small difference in the distributions will result in very small p-values. The p-value from the test will often be very small due to the size of the samples and the differences between simulations caused by the chaotic nature of the system. This makes it difficult to use the KS test to say whether the samples are from the same distributions. However, the KS test can still be used to relatively compare the similarity of the model and VelVer speed-up simulations to pure simulations. Two-sample KS tests also lose the nuance gained by examining distributions as they are again boiled down to a single number. This makes it easy to get an overview of how similar all the distributions of angles and bond lengths are, but it becomes difficult to examine a single distribution in detail. Because of this, they are only able to tell you the similarity of two distributions, not how they differ. The manual work of comparing distributions is therefore still needed.[38]

A way to help visually compare different distributions is by using quantile-quantile (Q-Q) plots. They plot the values at which pre-decided quantiles occur in different samples against each other. The more similar the distribution of the two samples is the more linear the plotted points will be. This can make it easier to visually evaluate the similarity of the distribution of a single angle or bond length of different samples but does not give a general overview like the KS test can.[39] In the end, evaluating the distributions of angles and bond lengths gives a more accurate view of the similarity in angles and bond lengths between different simulations than looking at average values. However, it is still bad at describing the overall geometry and properties of the system as you are still just looking at a single angle or bond length at a time.

3.1.3 Frequency of Changes to Bond lengths and Angles

Instead of looking at the absolute values of angles and bond lengths, one can also look at their rate of change throughout the simulation and find the frequencies of rotations and vibrations of the system. This was also one of the metrics used by R. Han and S. Luber to evaluate the performance of their ML training method.[37] To do this first the change in angles or bond length over the trajectory was calculated

$$\dot{a} = \begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_t \end{bmatrix} - \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{t-1} \end{bmatrix} \tag{37}$$

where a is the value of an angle or bond length at different time steps and $\dot{a}$ is the changes in that value between the time steps. The frequencies of these changes can be found using the Fourier transform. The Fourier transform was applied to $\dot{a}$ using Scipy's function fftfreq with a sample spacing of 0.5 fs, or the slightly increased time step for the VelVer speed-up, to match the time steps of the simulations. This yields the frequencies at which the angles and bond lengths of the system change which can be compared across different simulations. Just like with the average values and distributions looking at the frequencies does not give a complete picture of the geometry and properties of the system. However, it can still be used to compare the accuracy of the model and VelVer speed-up in describing this aspect of the system.

3.1.4 Root Mean Square Deviation

Root mean square deviation (RMSD) is a method of comparing the difference in the positions of atoms between two configurations of the same system. RMSD was one of the primary methods by Brutovsky *et al.*[19,20] to evaluate their model which used auto-regression to predict the non-bonding forces. As is evident from the name, RMSD calculates the root of the mean of the squared deviation between the same atoms in the two configurations. This means that the position of oxygen in one ethanol structure is compared to the positions of the same oxygen atom in another structure and so forth. RMSD can be calculated using the following formula:

$$RMSD = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(a_i - b_i)^2} \tag{38}$$

where a and b are the position vectors of atoms of the structure to be investigated and the reference structure respectively. To get relevant information out of RMSD it is important to choose the right reference structure. As I want to examine how the system evolves I decided to use the initial positions of the atoms in each simulation as my reference. I then calculated the RMSD between that reference configuration and the configuration of each subsequent time step in the simulation. This results in a number for each time step that describes how much the structure has moved from its original position. This can then be used to compare how different simulations change over time.

There are two contributing factors to the RMSD of any given time step; the overall drift of the molecule and the small oscillations of the atoms. The overall drift of the system dominates the overall trend of the RMSD while the oscillations of the atoms will result in small oscillations of the RMSD. The effect the drift has on the RMSD can be minimized by either translating the molecule to keep the center of mass constant throughout the trajectory or by aligning the trajectory using a tool such as VMD.[40] Keeping the center of mass constant removes the overall drift from RMSD but keeps in the effect of rotations of the molecule. Aligning the trajectory removes both the overall drift and rotation of the molecule leaving only the oscillations of the atoms in the RMSD. In this thesis, neither method was applied and the RMSD was calculated from trajectories which included the drift of the molecule. As the initial momenta of the system has a great impact on the initial movements of the molecule, and therefore the RMSD, the model cannot be compared to a simulation with a different seed. This is because the difference in RMSD between two simulations with different seeds is so large that they cannot be compared to each other.

RMSD does not describe the overall geometry of the molecule, but instead describes the molecule's movements over time. In addition to this, because RMSD is an aggregate of all the deviations of atoms in a system, two different structures can have the same RMSD. This means that when using RMSD as a metric, one must compare how it changes over time for the different simulations instead of using snapshot values. Because simulations with different seeds cannot be compared the RMSD cannot be used to evaluate if the model gives a valid solution to an *ab initio* simulation. Instead, the model and Velver speed-up simulations with the same seeds must be relatively compared to each other. The one of the two that has its RMSD closest to the pure simulation is then the most accurate way, in terms of RMSD, of speeding up the simulation. RMSD is a good metric to identify which simulations are relatively most similar to a target simulation. When a pure simulation is chosen as the target simulation RMSD can be used to relatively rank

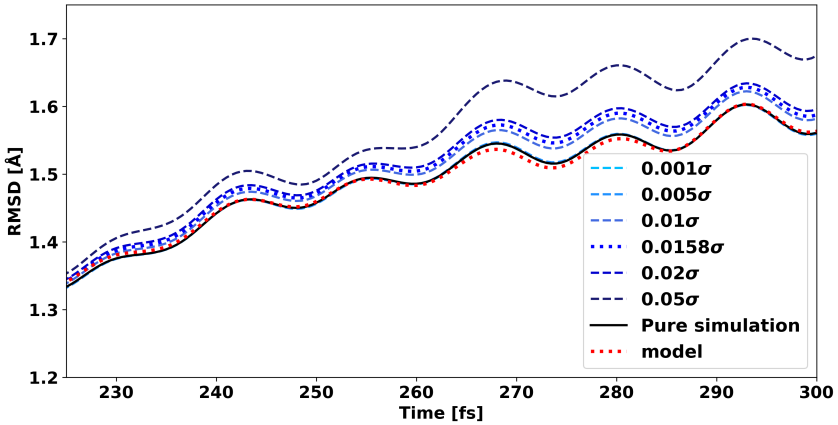


Figure 6: The RMSD of simulations with larger induced error is further away from the pure simulations than those with smaller induced error. The blue lines show the RMSD of simulations with a Gaussian noise of varying σ added to the DFT forces every 12 steps. The black line is a pure simulation and the red dotted line is the model. The dotted blue line has the same MAE as the model. The RMSD from the VAR_mol model is much closer to the pure simulations than inducing a similar error from a Gaussian distribution.

the accuracy of different simulations. This can be seen in Figure 6 which plots the RMSD of a model and VelVer simulations, as well as the RMSD of simulations where a random Gaussian error with mean 0 and different σ is added to the DFT calculated forces every 12 steps. The two simulations with an induced error of $\sigma = 0.001$ and $\sigma = 0.005$ follow the pure simulation almost perfectly at the shown time window. For the simulations with larger σ they are ordered so that the simulation with the largest induced error is the furthest away from the pure simulation. When comparing the the induced Gaussian error with the model, a Gaussian distribution with $\sigma = 0.0158$ corresponds to inducing an MAE equal to the model's. It can be seen in Figure 6 that the RMSD of the model is much closer to the RMSD of the pure simulation than if a random Gaussian error with comparable MAE was induced. This indicates that the model performs better than introducing random noise to the DFT calculated forces. Based on all this RMSD is a good metric to compare the relative performance of different models. However, it cannot be used to determine whether the geometry or properties of the system are kept intact throughout the simulation as other metrics can.

3.1.5 Conformational Isomers

There are two different conformational isomers that ethanol can take. It can either be in the anti configuration, where the hydrogen in the OH group points away from the carbon chain, or the gauche configuration, where the same hydrogen points towards the carbon chain.[41] The two conformers can be seen in Figure 7. The distribution of the conformers in model simulations and pure simulations can be examined to determine whether the model changes the energy landscape of the system. The rates of the conformers can also be compared to experimental data. However, in my experience, the rates vary depending on the starting parameters of the simulations, as can be seen later when the results are discussed. It will require a very large amount of data before the chaotic nature of the system is averaged out and comparisons to experimental data can be made. It is therefore better to compare the distributions of isomers in the model to VelVer speed-up and pure

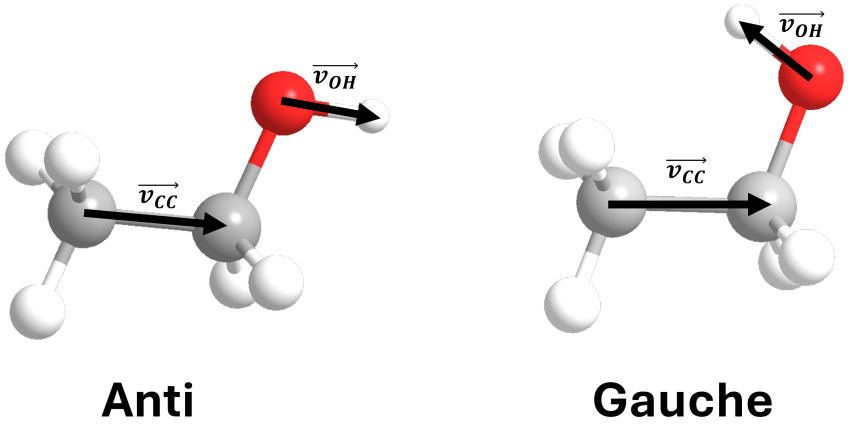


Figure 7: The two conformers of ethanol are shown with the gauche conformer on the left and the anti conformer on the right. An example of the vectors used in calculating the scalar vector projection is also shown.

simulations.

However, before the distributions of the conformers can be examined there needs to be a method to evaluate which conformational isomer the structure at a given time step is in. There are different ways of doing this but in my thesis, I did it by using scalar projection. In order to do this two vectors were created. One of the vectors, $\overrightarrow{v_{OH}}$ goes from the oxygen to the hydrogen atom in the OH group, while the other vector, $\overrightarrow{v_{CC}}$ spans the two carbon atoms and points towards the OH group. An illustration of the vectors can be seen in Figure 7. The scalar projection of $\overrightarrow{v_{OH}}$ vector onto $\overrightarrow{v_{CC}}$ was then calculated

$$s = \overrightarrow{v_{OH}} \cdot \frac{\overrightarrow{v_{CC}}}{\|\overrightarrow{v_{CC}}\|}. \tag{39}$$

In the case of the ethanol molecule being in the anti configuration the scalar projection, s, will be positive. If ethanol is instead in the gauche position the scalar projection will be negative. This creates an easy way to divide all time steps into the two conformers. Another way of dividing the molecule into its conformers is by calculating two RMSD values of the structure each time step. One of the RMSD values will be calculated with a reference structure in the anti configuration and the other RMSD value will use a reference structure in the gauche configuration. The reference structure with the lowest RMSD is then the isomer that ethanol is in for that time step. However, to do this requires that the trajectory is aligned such that translations and rotations of the molecule have no effect on the RMSD value calculated. This results in the configuration of the molecules being the only contributor to the RMSD. It is quite tedious to align all the trajectories one wants to examine and the scalar projection method was therefore used in this thesis.

Compared to the previous metrics discussed the distribution of conformers is better at both describing the overall geometry and properties of the system. It is important for many systems that the distribution and ratio of their isomers are explained correctly. In addition to this, the distribution of the conformers also evaluates whether the energy landscape of the system is preserved by the model. All of this means that the distribution of a system's conformers is the most important metric discussed so far. However, it is not the case that all systems have conformational isomers which means the metric cannot be used to evaluate the model's performance on all systems.

3.1.6 2D Dihedral Angles

Probably the most widely used metric used to evaluate models is 2D plots of dihedral angles.[22,37,42,43] Dihedral angles are better than angles at describing the geometry of a section of the molecule and by using two dihedral angles a large part of a system's geometry can be described. In addition to this by looking at two dihedral angles at the same time, it can be ensured that they occur in the same combinations across different simulations. 2D dihedral angles are often used to evaluate force fields by comparing the energy from the force field with the energy from DFT at each combination of two chosen dihedral angles. This way it can be assured that the model creates an accurate energy landscape of the system. Another advantage of using dihedral angles is that they are also good at describing the geometry of proteins expanding the use cases of the metric. This all makes it so that 2D dihedral plots are the most accurate way of evaluating the performance of a model as long as relevant dihedral angles can be chosen, such that they describe a large part of the system's geometry.

As my VAR_mol model does not predict the energy of the molecule I cannot create a heatmap of the energies at different dihedral angles. Instead, I created 2D histograms of the dihedral angles. This creates a population map that can be used to approximate the energy landscape of the system given enough data. The dihedral angles that I used were one spanning H-C-C-H, describing the carbon chain, and one spanning H-O-C-H which describes how the OH group is positioned relative to the carbon chain.

3.2 Results

In the following section, I will use the previously discussed metrics to evaluate the model's performance. For the VAR_mol model to be good enough to be used as a method to speed up simulations there are two criteria it must fulfill:

1. It must yield a valid solution. This is checked by comparing if the differences between a model and a pure simulation, with the same seed, are smaller than the difference between two pure simulations with different seeds.

2. The model must perform better than VelVer speed-up by being closer than it to a pure simulation with the same seed.

In the results section data from four different simulations will be used. There are three simulations with seed 1; the model, with 21069 total time steps, VelVer speed-up, with 22526 total time steps, and a pure simulation with 14756 total time steps. The last simulation from which data is used in this section is a pure simulation with seed 2 which has 14965 total time steps.

3.2.1 Average Values

The average values for a selection of angles and bond lengths can be seen in Table 2. A wider selection of the average values angles and bond lengths, including data from all four tested models, can be seen in Appendix C. The average values gained from a VAR_mol model with seed 1 are generally closer to a pure simulation with seed 1 than the average values from a pure simulation with seed 2. There are a few average values where this is not the case but these mostly occur for average values where there is a comparatively large variance between different simulations. This makes it so that those average values

Table 2: Average values of angles and bond lengths.

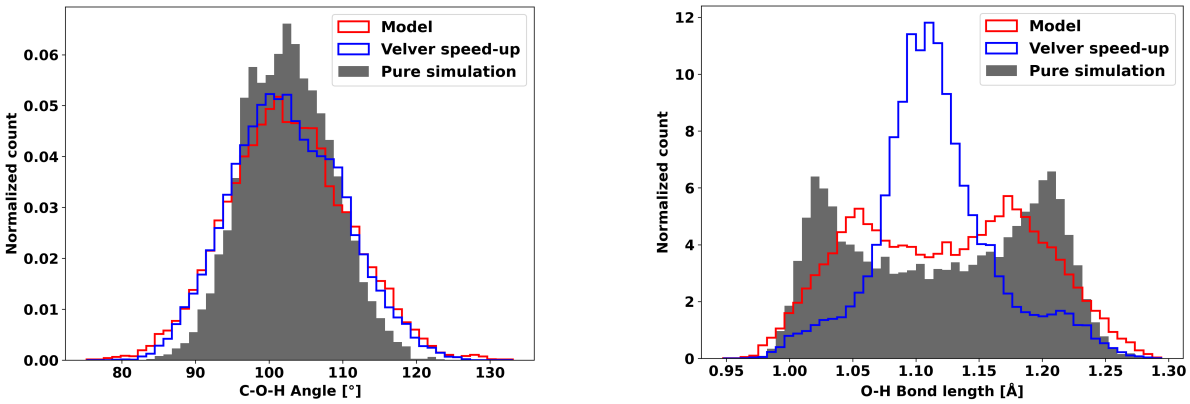
Seed	Simulation	H-C-C	H-C-O	C-O-H	C-C	O-H
	Pure simulation	110.74°	112.54°	102.57°	1.614 Å	1.121 Å
1	Model	110.45°	110.79°	102.61°	1.621 Å	1.124 Å
	Velver	110.54°	111.11°	102.52°	1.616 Å	1.116 Å
2	Pure simulation	110.41°	111.94°	102.86°	1.605 Å	1.125 Å

are more sensitive to differences introduced by the initial momenta or the methods used in the simulation. These values will therefore also have a higher spread of valid solutions. When comparing the model to VelVer, it can be seen that the average values from VelVer are generally slightly closer to those from pure simulations than those from the model are. However, for the average O-H bond length, the model is closer than VelVer to the pure simulation. In addition to this, the average C-O-H angle is also more accurately described by the model.

Based on the average values of the different *ab initio* MD simulations done, the model seems to be within the range of what is expected from a valid solution. This is especially the case when considering that the model speeds up the *ab initio* MD when compared to the pure simulations, and it is therefore expected that the model will introduce some errors. VelVer seems to give more accurate average values for most angles and bond lengths in the system than the Model. However, the model gives more accurate values for angles and bonds which includes the whole hydroxyl functional group. This indicates that VelVer is better at describing the main carbon chain while the model is better at describing the hydroxyl group.

3.2.2 Distributions

As was mentioned in the metrics section not a lot of stock should be put in average values as different distributions can have the same average value. This can be seen in Figure 8 which shows the distributions of the C-O-H angle and the O-H. The distribution of the C-O-H angle in Figure 8a for the VAR_mol model and VelVer closely resembles each other. The overall shape of both distributions is similar to the shape of the distribution from the pure simulation. However, the model and VelVer distribution are a bit wider than the pure simulation. This matches the information gained from the average value fairly well as it is obvious that neither the model nor VelVer perfectly replicates the pure simulation. However, it is not obvious from the distribution that the model is more accurate than the VelVer, which was indicated by the average values. For the O-H bond length, the average value from the VelVer simulation indicated that it only performed slightly worse than the model. However, when we look at the distributions of the bond length in Figure 8b we can see that there is a double peak in the pure simulation's distribution. This double peak is most likely due to ethanol's two conformers, which result in different O-H bond lengths. Although the difference between the two peaks is larger than what is expected.[41] VelVer fails to exhibit this double peak in its distribution. On the other hand, the distribution from the model does exhibit this double peak although it still varies noticeably from the pure simulation. Looking at the distribution of the O-H bond length reveals that VelVer completely fails to replicate an aspect of the system that the model does replicate. This would have been missed if average values were used as the only metric to evaluate the simulations.



(a) (b)

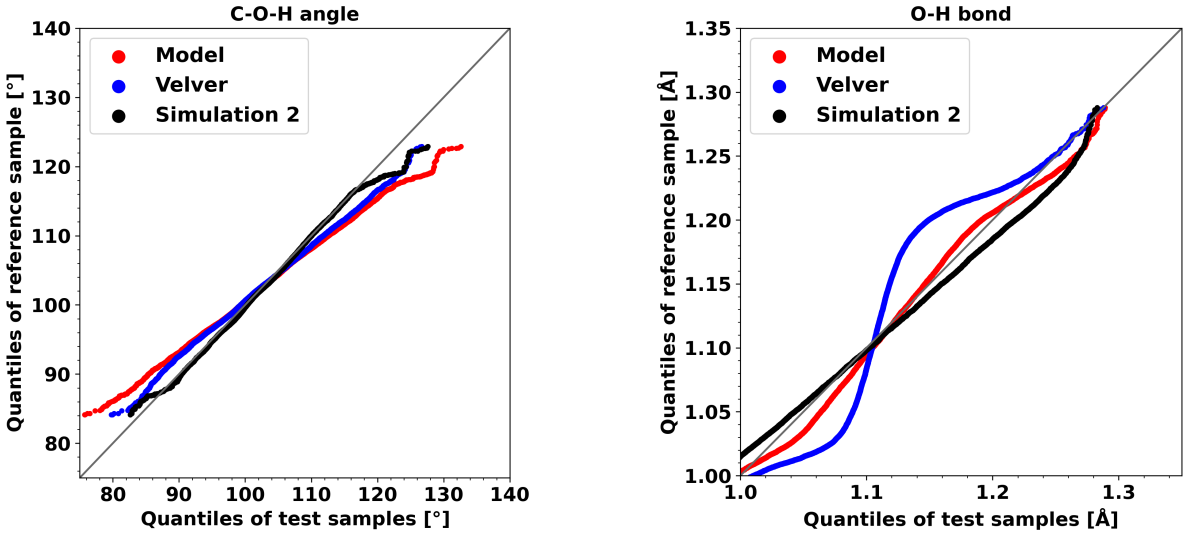
Figure 8: The VAR_mol model and Velver similar distribution of the C-O-H angle. For the O-H bond length, the distribution of the model matches the pure simulation much better than VelVer which fails to replicate the double peaks. a) shows the distribution of the C-O-H angle. b) shows the distribution of the O-H bond length. Both plots show data from a simulation, the model, and VelVer speed-up all of which were run with the same seed.

Table 3: P-values from KS test of angles and bond lengths.

Seed	Simulation	H-C-C	H-C-O	C-O-H	C-C	O-H
	Pure simulation	1	1	1	1	1
1	Model	$1.2 \cdot 10^{-4}$	$4.8 \cdot 10^{-102}$	$3.7 \cdot 10^{-45}$	$5.3 \cdot 10^{-29}$	$1.9 \cdot 10^{-45}$
	Velver	$6.6 \cdot 10^{-6}$	$2.6 \cdot 10^{-78}$	$6.8 \cdot 10^{-35}$	$3.2 \cdot 10^{-7}$	0
2	Pure simulation	$1.2 \cdot 10^{-36}$	$5.5 \cdot 10^{-38}$	$6.3 \cdot 10^{-7}$	$2.4 \cdot 10^{-36}$	$5.7 \cdot 10^{-80}$

To get an overview of the similarity of the different angles and bond lengths across different simulations the two-sample KS test was used. The pure simulation with seed 1 was always used as a reference sample and the KS test therefore shows how similar the other simulations are to this one. The p-values from two-sample KS tests of a selection of angles and bond lengths can be seen in Table 3. The p-values from the two-sample KS tests of all models on a wider selection of angles and bond lengths can be seen in subsection D.1. Here it can be seen that VelVer performs similarly or better than the VAR_mol model in almost all angles and bond lengths. For the C-O-H angle, the KS test shows that VelVer is slightly better than the model which makes sense given Figure 8a but contradicts what was found with the average values. The KS test also gives VelVer a 0 for the O-H bond distribution which shows that VelVer completely fails to replicate the pure simulation in this aspect. It can therefore be seen that the KS test gives a single number that can compare the whole distribution instead of just the mean of the distribution. Because of this, the KS test gives a more accurate evaluation of the performance of the model than looking at the average values. The similarity between pure simulations with different seeds varies greatly depending on what angle or bond length is examined. For some, the model performs better than a pure simulation with a different seed although for others the model performs worse. This makes it difficult to evaluate whether the model yields a valid solution based on the distribution of angles and bond lengths.

One disadvantage of the KS test is that it is difficult to understand what the difference



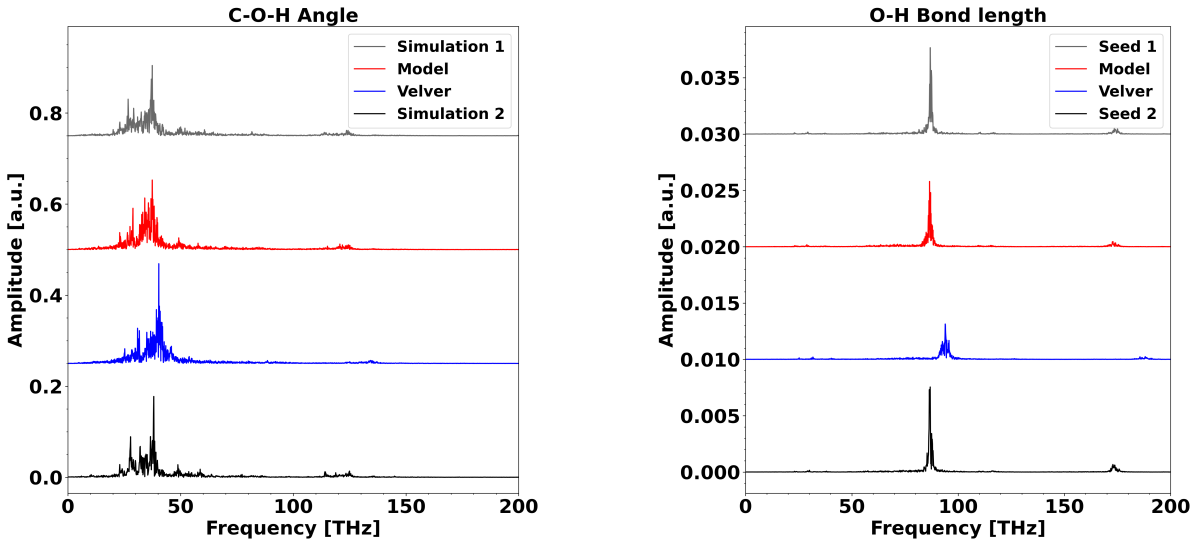
(a) (b)

Figure 9: a) shows a Q-Q plot of the C-O-H angle and b) shows a Q-Q plot of the O-H bond length. A pure simulation with seed 1 is used as the reference sample in both plots. For the C-O-H angle a pure simulation with seed 2, black line, is closer to the reference than seed 1 simulations of the model, red line, and VelVer speed-up, blue line. For the O-H bond length, the VAR_mol model and the simulation with seed 2 are about equally close to the reference sample while the VelVer is noticeably further away.

in p-values means for the accuracy of the simulations. For example. for the C-O-H angle it can be seen in Figure 8a that the differnce between the model's p-value of $3.7 \cdot 10^{-45}$ and VelVer's p-value of $6.8 \cdot 10^{-35}$ is small, but how does that compare to the pure simulation with seed 2's p-value of $6.3 \cdot 10^{-7}$? If the distributions of all simulations were plotted as histograms it would be chaotic and difficult to compare them. A better way of evaluating how different p-values transition to differing distributions is by using Q-Q plots. Q-Q plots of the model, VelVer, and pure simulation with seed 2 compared to the pure simulation with seed 1 can be seen in Figure 9. Q-Q plots with all tested models can be seen in subsection D.2. In Figure 9a it can be seen that for the C-O-H angle, the VAR_mol model and VelVer perform similarly with VelVer being slightly better. The pure simulation with a p-value of $6.3 \cdot 10^{-7}$ performs noticeably better than the model and VelVer. Figure 9b once again shows how VelVer's distribution of the O-H bond length is fundamentally different from the distributions of the other simulations.

3.2.3 Frequency of Changes to Angles and Bond lengths

The dominating frequencies for the change of the C-O-H angle and O-H bond length can be seen in Figure 10. The frequencies of the change of the C-O-H angle for the different simulations in Figure 10a are very similar. The main group of peaks is in the $25\,\text{THz} - 50\,\text{THz}$ range with there being another small group of peaks around 125 THz. The biggest difference between the different simulations is that the largest peak in the VAR_mol model seems to be a bit broader than it is in the other simulations. Other than that the small group of peaks occurs at a slightly higher frequency for the VelVer simulation than the others. However, there is a larger difference in the frequencies of the change in the O-H bond as can be seen in Figure 10b. Here the two pure simulations



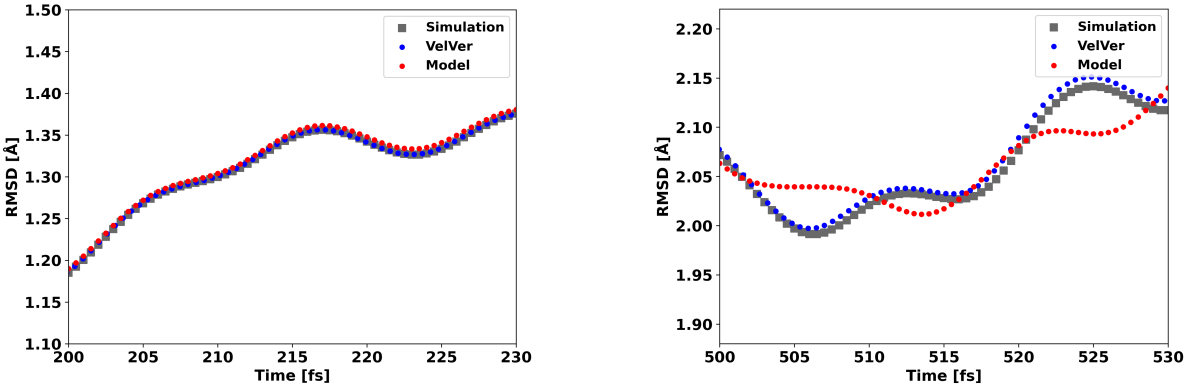
(a) (b)

Figure 10: a) shows the amplitudes of the Fourier transform for the change in the C-O-H angle. b) shows the amplitudes of the Fourier transform of the changes in the O-H bond length. The frequencies for the C-O-H angle are similar across all simulations. For the frequencies of the O-H bond length, VelVer is different from the other simulations. Only the positive frequencies are shown and the amplitudes have been manually shifted to make it easier to compare the different simulations.

and the model are again very similar. However, the main peak of VelVer occurs at a higher frequency and is broader and shorter than in the other simulations. The same difference can be seen with the smaller group of peaks that occur around 175 THz in the two pure simulations and the model. Overall analysing the frequency of changes in angles and bond length corroborates the information gained in the previous metrics. The model gives a solution that is within the range of a valid solution based on the differences between the two pure simulations with different seeds. VelVer performs slightly better than the model when describing the main carbon chain of ethanol. However, the model describes the hydroxyl group better than VelVer.

3.2.4 Root Mean Square Deviation

The RMSD of a section of the pure simulation, VelVer, and VAR_mol model with the same seed can be seen in Figure 11. The RMSD of a section of all tested models can be seen in Appendix E. Simulations with different seeds cannot be compared using RMSD as the initial momenta of a system has a large impact on the RMSD. In Figure 11a it can be seen that the RMSD of the different simulations is very similar at the start of the simulations. The differences in the forces of the model and VelVer simulations have not accumulated enough to create a noticeable difference from the pure simulation. However, the VelVer is a little closer to the pure simulation than the model is. As was seen with Figure 6 the smaller the error is introduced in a simulation the closer it will be to the pure simulation. Based on this Figure 11a suggests that VelVer is better at describing the motion of the system than the model is. This is exacerbated in Figure 11b which shows the RMSD at a later point in time. Here the VelVer simulation still follows the pure simulation. The model roughly describes the overall drift of the molecule but the



(a) (b)

Figure 11: The RMSD of VelVer more closely matches that of the pure simulation than the RMSD of the VAR_mol model does. The RMSD is calculated between the molecule at each frame and the molecule in its original position in the first frame of the trajectory. a) shows a time frame where the RMSD of both the model and VelVer follow the RMSD of the pure simulation. b) shows a time frame where the model RMSD no longer follows the oscillations of the pure simulation RMSD. However, the overall drift of the molecule is still mostly replicated.

smaller movements in the system, like rotations and vibrations, no longer follow the pure simulation. The RMSD therefore shows that VelVer is better than the model at describing the overall movements of the system.

It should be noted that the RMSD describes the mean deviation of the molecule from, in this case, its initial positions. Even though the RMSD of VelVer is closer than the model is to the pure simulation it does not mean that VelVer describes all aspects of the system better than the model. The model can therefore still describe the hydroxyl group better than VelVer, as was seen based on the average values and distribution of the O-H bond length. The average values and distributions also showed that VelVer was better than the model at describing the rest of the system. Describing the main carbon chain of the system better, as VelVer does, is more important for the overall movements of the system than more accurately describing the hydroxyl group, as the model does, as it makes out a larger part of the system. The evidence from the previous metrics can therefore be consistent with the fact that VelVer better describes the overall movements of the system.

3.2.5 Conformational Isomers

The distribution of the scalar projection calculated using Equation 39 which describes which conformational isomer the state is in can be seen in Figure 12. The figure shows the scalar projections for two different seeds of the model, VelVer, and pure simulation. The distribution of the scalar projections of all tested models can be seen in Appendix F. There is a large difference in the distribution of the two pure simulations with different seeds. The VAR_mol model is closer to the pure simulation with the same seeds than the two pure simulations are to each other. However, there are aspects of the pure simulations that the model fails to replicate. The model generally has a distribution with a peak at each end, although the gauche peak in seed 2 is small, with a mostly flat valley in between. The pure simulations on the other hand have an undulation of smaller peaks between the

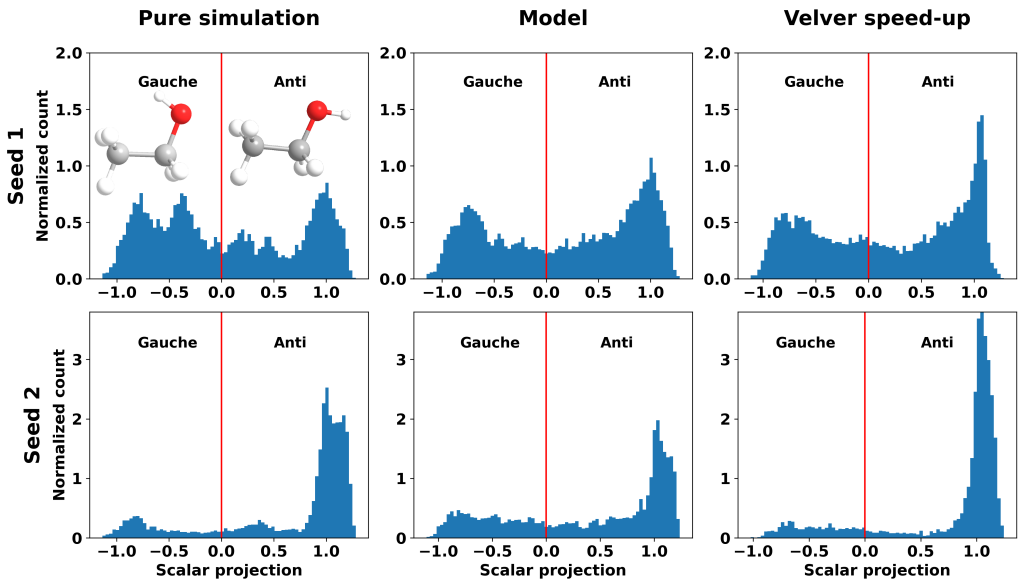


Figure 12: The distribution of the scalar vector projection, which describes the conformational isomer distribution of the simulation, of the VAR_mol model is closer than VelVer to the pure simulation with the same seed. There is a large difference in the distribution depending on the seed used and the difference between model and pure simulation is small in comparison. The scalar projection was calculated using Equation 39. Each column has different methods of simulation, while the rows are different starting seeds for each simulation method. There are three different simulation methods all performed on the same two seeds.

main peaks at each end. This is most easily seen in seed 1, but the seed 2 pure simulation still has more peaks than the seed 2 model. Even though the model does not describe the distribution with the same resolution as the pure simulation, it is described well enough that based on this metric the model yields a valid solution.

When the VAR_mol model is compared to VelVer, it is obvious that the model is closer than VelVer to the distribution of the pure simulation or both seeds. This is expected as the previous metrics examined showed that the model was better than VelVer at describing the hydroxyl group, which is what is examined with this scalar projection. However, VelVer does describe the pure simulations better than what could be feared based on its failure to replicate the double peak in the distribution of the O-H bond as seen in Figure 8b. VelVer consistently overestimates the amount of time the scalar projection is close to 1 in both seeds but replicates the overall shape of the distribution fairly well. Here it should be kept in mind that when evaluating the overall geometry of ethanol the distribution of conformers is more important than the distribution of the O-H bond length. Based on the scalar projection describing the conformers VelVer is still worse at describing the hydroxyl group than the model is but the difference between the two methods is much smaller than what the distribution of the O-H bond length showed.

3.2.6 2D Dihedral Angles

The 2D histograms of the dihedral angles describing the carbon chain and hydroxyl group of ethanol can be seen in Figure 13. It shows 2D histograms for 2 different seeds of the

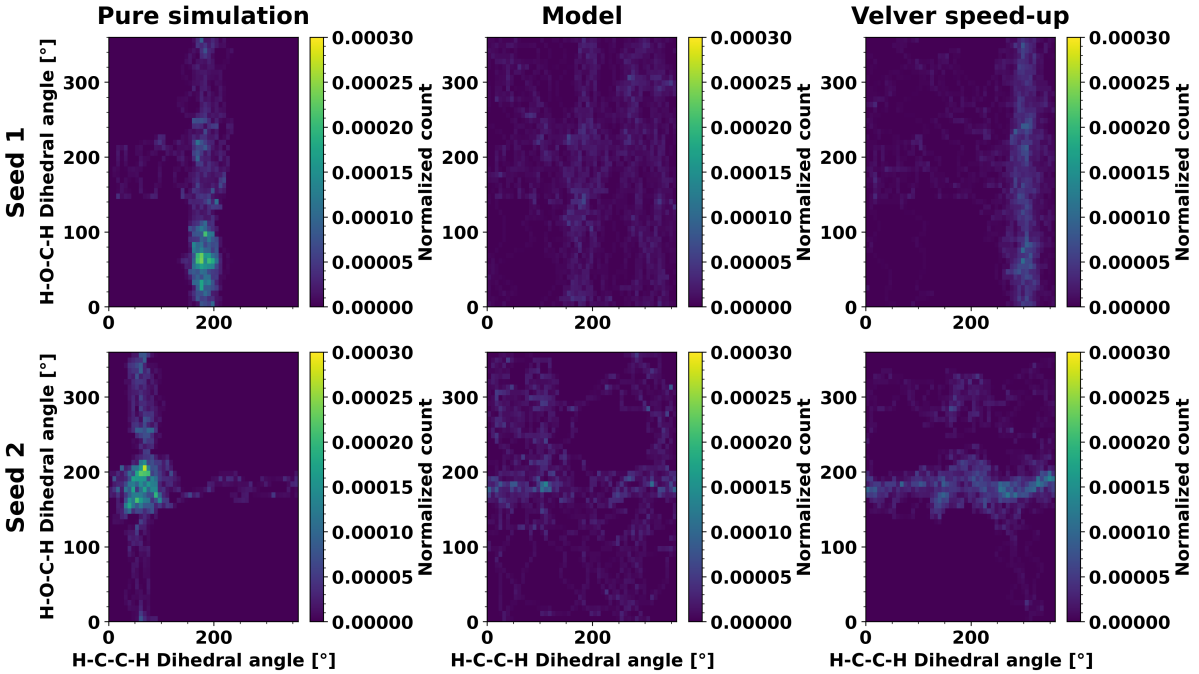


Figure 13: The histograms of the VAR_mol model are much more smeared out than the pure simulation histograms erasing any pattern present in those. The VelVer histograms more closely match the pure simulations as some of the pattern is kept intact, however, they are still more smeared out than the pure simulations. The figure shows 2D dihedral histograms of pure simulation, model, and VelVer for two different seeds. The difference due to methods can be seen horizontally while the difference due to seeds can be seen vertically.

model, VelVer, and pure simulations. The 2D histograms of all tested models can be seen in Appendix G. The 2D histograms of the pure simulations consist of a vertical and horizontal line. The horizontal line occurs at around 200° of the C-O-C-H dihedral while the exact location of the vertical line shifts around in the different pure simulations. For both pure simulations, most of the histograms are concentrated in a single blob which occurs at different locations for the two pure simulations. When looking at the histograms of the VAR_mol model there are almost none of the structures that are present in the pure simulations. Instead of having two lines the model's histogram is instead smeared out over all possible dihedral angles. Unlike the pure simulations, there are also no blobs that contain most of the population in the models. When looking at the 2D histograms of VelVer they are somewhere in between the model and the pure simulation. There are clear lines in the VelVer histograms but the population is smeared out, and not concentrated in the blobs seen in the pure simulations. The exact locations of the lines are also different when compared to the pure simulations with the same seed. Overall the large difference between the model and pure simulation, and the fact that VelVer does replicate some of the features in pure simulation, means that based on the 2D plots of the dihedral angles the model does not yield a valid solution to the *ab initio* MD. In addition to this, the model also performs noticeably worse than VelVer.

3.3 Perspective

One's evaluation of the performance of the VAR_mol model and whether it is better than VelVer heavily depends on the metrics one chooses to focus on. In the simple metrics, like angles and bond lengths, the model is within what a valid solution would be expected to be. VelVer would describe most of the system better than the model, but the model would be better at describing the hydroxyl group. This could then be corroborated by looking at the distribution of conformational isomers, where again the model is well within the range of a valid solution and performs better than VelVer. However, all of these metrics only look at one aspect of the system at a time. Even the more complex metric of the conformational isomer distribution only depends, for ethanol, on where the hydrogen in the hydroxyl group points. None of these metrics examines the overall geometry and properties of the system. Using metrics that examine the overall movement or geometry of the system, like RMSD or 2D dihedral angles, paints a different picture. In both of these metrics, the model performs markedly worse than VelVer. The model also does not give a valid solution when evaluated using 2D dihedral angles as the metric. The more complex metrics therefore show that the the model does not perform well enough to be used to speed up the *ab initio* MD simulation of ethanol. If one wants to speed up the simulation increasing the time step used in VelVer is better as it results in more accurate simulations and it is easier to do. One can argue that the model does describe the hydroxyl group better than VelVer. However, the small increase in the accuracy of the ratio of conformational isomers does not make up for the model completely failing to describe the overall geometry of the system as seen in the 2D histograms of the dihedral angles. This all highlights that it is important to use advanced metrics when evaluating the performance of a model. Using only simple metrics can give a wrong view of the performance of the model. Luckily examining 2D plots of dihedral angles is a widespread metric used in evaluating models but if a paper only uses simple metrics one should be critical of the results.

Ultimately the VAR_mol model introduces too many errors in an *ab initio* MD simulation of gas-phase ethanol compared to the speed-up gained from implementing the model. As the model seems to outperform random error as seen in Figure 6 the model does have some potential. In order to achieve that potential the question becomes if there are any ways to improve the implementation of the model. There are two general ways to do this. It can either be done by increasing the speed-up gained from implementing the model or by reducing the errors introduced. It is not realistic to increase the speed-up of the model as it already takes almost no time compared to calculating DFT forces, and according to the grid search choosing an input lower than the $t = 12$ used in the examined model will lead to a higher MAE of the model. The last option for increasing the speed-up of the model is to increase the number of time steps predicted. However, the MAE was shown to increase exponentially with time in Figure 4a and it can be seen in Figure 14 that increasing the number of time steps predicted only leads to the 2D histograms of dihedral angles being even worse. As there is no feasible way of increasing the speed-up gained from the model the only option is to reduce the error introduced by it.

Due to the chaotic nature of the system, it only takes one atom being out of position to get a completely different result. This is a problem for the VAR_mol model because even though its residuals are centered around 0, as seen in Figure 4b, when the model is applied many times at some point a large unlikely error is introduced. When this unlikely

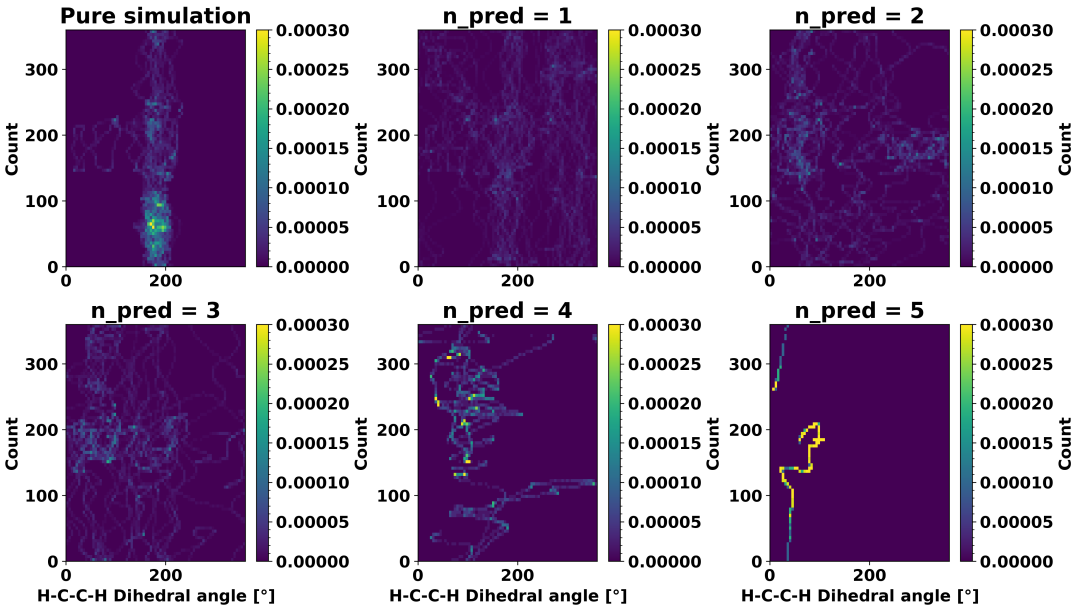
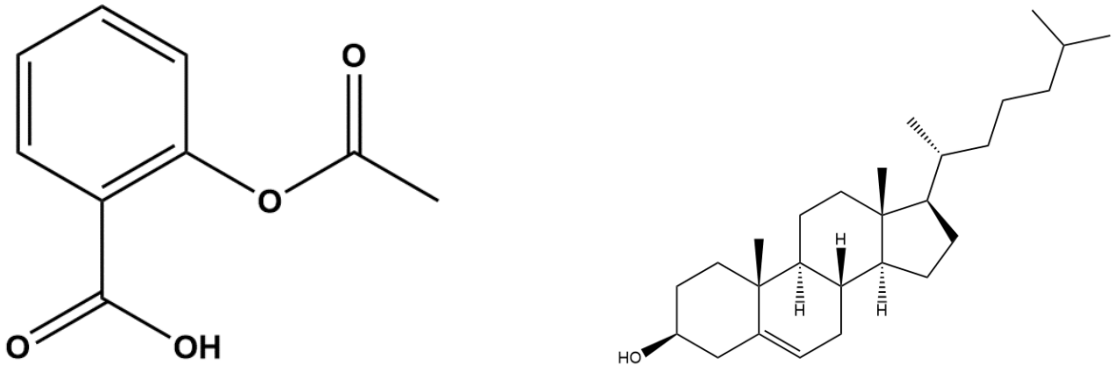


Figure 14: The figure shows 2D dihedral histograms of a pure simulation and models with an increasing number of predictions made. The histograms quickly break down. This results in any VAR_mol model that makes more than 2 predictions being unrecognizable and performing way worse than doing 1 prediction, which still performs badly compared to the pure simulation.

error is introduced at a time step it will be in 1 out of 27 force trajectories due to the small size of ethanol. This means that when a one-in-a-thousand error is produced by the model it will result in a large MAE for that time step as there are not enough other forces predicted to cancel out the unlikely error. In addition to this, the small number of atoms in ethanol means that one atom being knocked out of course has a large effect as the rest of the system is not large enough to reel it back into position. Because of this large errors introduced by the model will have a disproportionate effect on the simulation. One easy way to counteract this is just by using systems with a larger amount of atoms. This will create a larger buffer of atoms to counteract the one atom where a large error has been introduced by the model. Another reason why the model performed poorly on ethanol could be due to it having a varying energy landscape as it contains a hydroxyl group and only has a short carbon chain to act as a stable backbone. This will likely create more sudden changes in the forces experienced by the atom in the system, which in turn means that it will be more difficult for the model to predict these forces. This can again be counteracted by choosing a larger system with many of the same atoms in a predictable structure. This could either be large hydrocarbons or a material simulation, as they will have a very consistent energy landscape. The downside of making the model more stable by doing this is that it will also in general make the *ab initio* MD simulations more stable. This means that larger time steps can be used without affecting the simulation adversely which will increase the requirements for the model's performance. In section 4 I will explore how the model performs on heavier molecules.

Another way of improving the performance of the VAR_mol model is by adding information it can use. The VAR model might already be extracting and using all the information available in force trajectories which leaves the model in its current state, where it performs better than random noise but not good enough to be used in *ab initio*



(a) Aspirin (b) Cholesterol

Figure 15: a) shows the structure of aspirin while b) shows the structure of cholesterol.

MD simulations. One way to increase the information available is by using molecular descriptors which are different ways of numerically describing the chemical properties of a system.[44] However, molecular descriptors are not directly related to the forces and are too complicated to be used in a VAR model so another type of model is needed to make use of them. This could be a recurrent neural network (RNN) which is an ML model that is especially suited to predict data in a time series.[45] The goal of creating an RNN is that it can have the same advantages as the VAR model, low start-up cost and general applicability, while reducing the error by incorporating information on the chemical environment from descriptors. In section 5 I will do a short exploration of using RNN to predict molecular forces.

4 Applying the Model on Heavy Molecules

As mentioned in the perspective section the VAR_mol model may perform better when applied to larger systems. In order to test this the model was applied to two different systems consisting of relatively heavy molecules and the resulting simulation was evaluated. The two molecules that were chosen to test this are aspirin and cholesterol, both in the gas phase. The structure of aspirin and cholesterol can be seen in Figure 15. Aspirin was chosen as a molecule that was heavier than ethanol but still had a varying energy landscape due to the many oxygen atoms present. Cholesterol was chosen as a best-case scenario for the model. It is a relatively heavy molecule and it consists mainly of carbon and hydrogen atoms. This will create both a large buffer of atoms to make sure the rare large errors do not have a big effect on the system and a consistent energy landscape which will make it easier for the model to predict forces.

As 2D dihedral angles were the best metric for evaluating the model's performance, and the metric the VAR_mol model failed to replicate for ethanol, this will be the main metric used in this section. There are however some difficulties in applying this metric to heavy molecules. As mentioned in the metrics section, the model cannot calculate the energy of the molecule, so a population map is instead used as a proxy for a heatmap of the energies at the different dihedral angles. However, a certain amount of data points is required for the population map to be a valid proxy. As aspirin and cholesterol consist of significantly more atoms than ethanol *ab initio* MD simulations of them will also take

longer. This will naturally result in fewer data points being available. However, the 2D histograms can still be used to compare the combinations of dihedral angles across the different simulations. Another shortcoming of analyzing the 2D dihedral angles for heavy molecules is that it will no longer describe the geometry of the whole molecule like it was able to do for ethanol. Instead one has to choose dihedral angles that describe certain areas of interest in the molecule. The 2D dihedral angles can then be used to analyze the geometry of the molecule in these areas. However, in many cases, dihedral angles can be chosen such that they describe the most important aspects of the molecule. Taking all of this into account 2D dihedral angles are still the best metrics in describing the geometry of the molecule.

The *ab initio* MD simulations of the heavy molecules were done using the same settings as described for ethanol in subsection 2.2, except for a temperature of 600 K being used in the Maxwell-Boltzmann distribution. A VAR_mol model was used to forecast both of the heavy molecules and the hyperparameters were optimized using a grid search.

4.1 Aspirin

For Aspirin a VAR_mol model with $t = 15$, $p = 3$, and $\lambda = 0$ was used to forecast the forces. Just like with ethanol, three different kinds of simulations were done. These are pure simulations only using DFT, model simulations incorporating the forecasted forces into the simulation, and VelVer simulations which increase the time step to achieve a similar speed-up to the model. The time step needed to do this was calculated using Equation 33

$$\Delta t_{speed-up} = \frac{16}{15} \cdot 0.5\,\text{fs} \approx 0.533\,\text{fs}. \tag{40}$$

These three kinds of simulation were each performed on the same two seeds. Each simulation consists of 5900 time steps, which is fewer than the ethanol simulations due to the increased size of the system simulated. The two dihedral angles that were used in the 2D histograms were chosen so they described the orientation of the ester and carboxylic acid functional groups compared to the carbon ring. For the ester, this is done by calculating the dihedral between two carbon atoms on the ring, and the oxygen and carbon atoms on the ester. The dihedral angle for describing the ester is C-C-O-C where the two first carbon atoms are on the carbon ring. The next oxygen and carbon atoms are from the ester. For describing the carboxylic acid the dihedral angle used is C-C-C-O where the first two carbon atoms are again on the carbon ring. The third carbon is in the carboxylic acid and the oxygen is from the OH group in the carboxylic acid.

The 2D histograms of the dihedral angles of aspirin can be seen in Figure 16. The histograms of the pure simulations roughly take the shape of a circle with its center at 50° of the ester dihedral and 170° of the carboxylic acid dihedral. When looking at the 2D histograms of the VAR_mol model and VelVer both of them also roughly take the shape of a circle with the same center as the pure simulation. When comparing the shapes the model seems to be more concentrated in the same or slightly smaller area than the pure simulation. VelVer on the other hand seems to be less confined and extend more outside the area of the pure simulation. In addition to this, the pure simulation of seed 1 seems to follow the diagonal of the plot. This diagonal aspect of the 2D histogram is not replicated by either the model or VelVer. It is also not replicated by the pure simulation with seed 2.

As the differences between the VAR_mol model and VelVer are so small, it is difficult

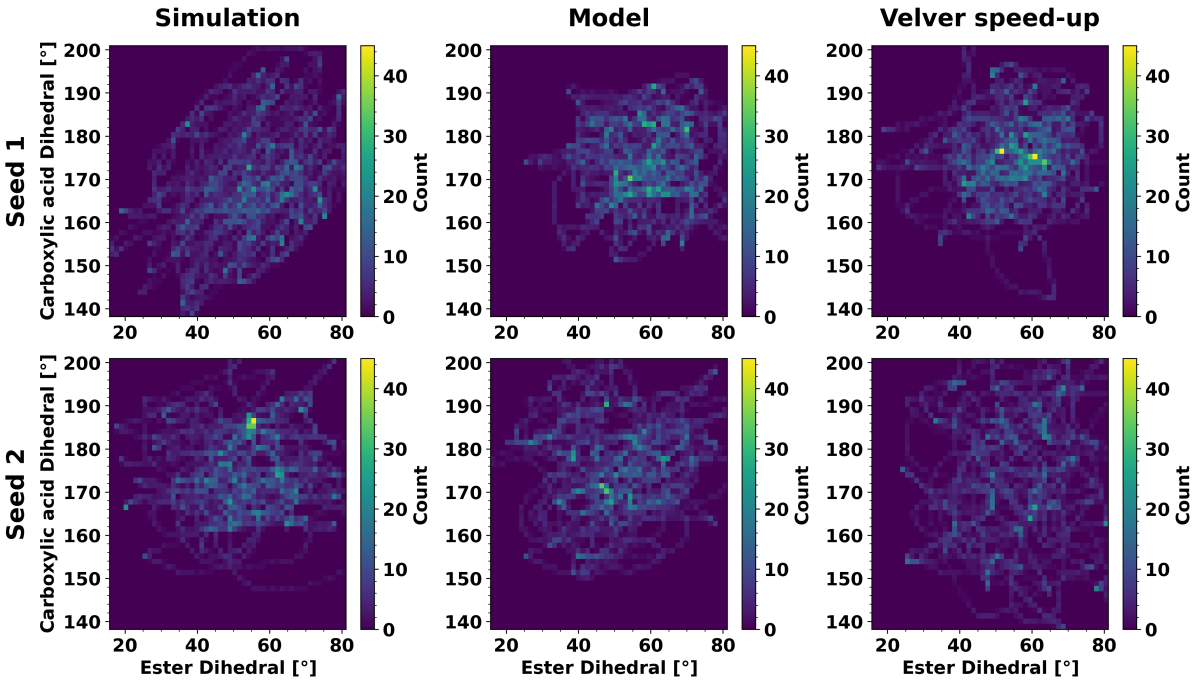


Figure 16: The 2D histograms of aspirin generally look like circles. The 2D histograms of the VAR_mol model roughly stay within the area of the pure simulation. The VelVer histograms seem to be centered around the same place but are less constrained and sometimes extend outside of the area of the pure simulation. The figure shows 2D dihedral histograms of pure simulation, model, and VelVer for two different seeds. The difference due to methods can be seen horizontally while the difference due to seeds can be seen vertically.

Table 4: The RMSD of aspirin's 2D dihedral

	Seed 1	Seed 2
Model	4.41	4.13
VelVer	4.63	4.16

to evaluate which one performs the best by eye. In order to help with this the RMSD of the model and VelVer histograms compared to the pure simulation histogram are calculated. This will give a single number that will show how much the model and VelVer deviate from the pure simulations. The calculated RMSD values can be seen in Table 4. Here the RMSD of the model is slightly lower than the RMSD of VelVer. However, it can still be seen that they perform similarly as the difference between the two, especially in seed 2, is small.

Overall the difference between the different seeds of the pure simulations is larger than the difference between the VAR_mol model and pure simulation meaning the model yields a valid solution to the *ab initio* MD simulation. The model is also slightly better than VelVer at replicating the pure simulation. All of this is a stark improvement when compared to the performance of the model on the ethanol system. With that being said the differences between the model and VelVer are small. When combined with the smaller sample size of the simulations it is difficult to say whether the extra hassle of using the VAR_mol model, compared to just increasing the time step in VelVer, is worth a possible small gain in performance.

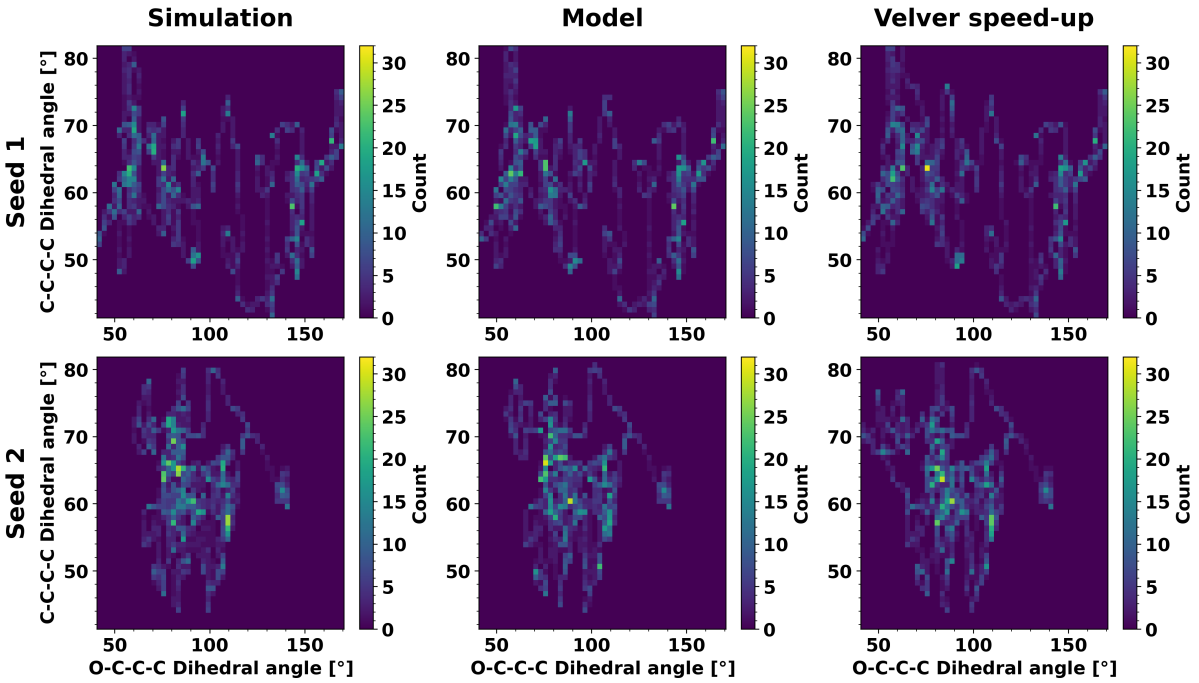


Figure 17: The 2D dihedral histograms of cholesterol vary a lot depending on the seed. The VAR_mol model and VelVer perform similarly, but the model seems to describe the upper left quadrant of the histograms better than VelVer. The figure shows 2D dihedral histograms of pure simulation, model, and VelVer for two different seeds. The difference due to methods can be seen horizontally while the difference due to seeds can be seen vertically.

4.2 Cholesterol

The hyperparameters of the VAR_mol model used to forecast the forces of cholesterol were $t = 16$, $p = 3$, and $\lambda = 0$. Just like with aspirin, pure simulations, simulations incorporating the model, and VelVer simulations with two different seeds were done. The time step used by VelVer to achieve a comparable speed-up to the model was calculated using Equation 33

$$\Delta t_{speed-up} = \frac{17}{16} \cdot 0.5\,\text{fs} \approx 0.531\,\text{fs}. \tag{41}$$

As cholesterol is a larger system than both ethanol and aspirin the simulations only consist of 2800 time steps. Due to the large size of cholesterol, it is also difficult to choose dihedral angles that describe the overall geometry. This is made even more difficult by the fact that, unlike aspirin, cholesterol does not have any clear functional groups that could be used. The first dihedral used was C-C-C-O to describe the hydroxyl group. The oxygen is in the hydroxyl group and the three carbon atoms are in the carbon ring which the hydroxyl group is attached to. The second dihedral angle is C-C-C-C, which describes how the long carbon chain is attached to the group of carbon rings. It uses the two first carbon atoms connected to the carbon chain. The other two carbon atoms are in the carbon rings with the first being the atom that the carbon chain is connected to. The last carbon atom is the one attached to the connection point and that points towards the rest of the carbon rings.

The 2D histograms of the dihedral angles of cholesterol can be seen in Figure 17. There is a very large difference between the two pure simulations with different seeds. Compared

Table 5: The RMSD of cholesterol's 2D dihedral

	Seed 1	Seed 2
Model	1.94	2.09
VelVer	1.50	2.05

to this difference, the VAR_mol model is much closer to the pure simulation with the same seed and is therefore a valid solution. Just like with aspirin the 2D histograms of the model and VelVer are very similar. However, for both seeds, the model does visually seem to describe the upper left quadrant more accurately than VelVer. The RMSD values of the model and VelVer compared to the pure simulations can again be used to help quantify their performance. The RMSD values can be seen in Table 5. Here the RMSD values of VelVer are smaller than the model's, although the differences between the two are minor in seed 2. This means that while it visually seems that the model performs better than VelVer, when the deviation from the pure simulation is quantified VelVer is closest. VelVer therefore seems to perform slightly better than the model, but it is difficult to be certain of this due to the low sample size. In Figure 17 one can see the strings of the paths through the two dihedral angles that the molecule took throughout the simulations. In addition to this due to the large size and rigidity of cholesterol, two dihedral angles cannot describe the overall geometry of the system. This all makes it difficult to be certain of the accuracy of the evaluation.

4.3 Perspective

Overall applying the VAR_mol model to heavy molecules shows promise. The model yielded valid solutions both for aspirin and cholesterol. The model performed about as good if not slightly better than VelVer for aspirin and slightly worse than VelVer for cholesterol. Based on this the hypothesis that the model performs better on systems with more atoms seems true, as the model performed much better on both heavy molecules than it did on ethanol. However, the theory that the model would perform better on systems with a consistent energy landscape is not correct. The fact that the VAR_mol model performs better on aspirin and cholesterol than on ethanol opens an avenue for future use cases of the model as it could be used for simulation consisting of heavy molecules. However, the low sample size used to evaluate the model's performance on aspirin and cholesterol brings with it a degree of uncertainty. Also, as the model at best only performed slightly better than VelVer more research is needed to confirm that it is a possible avenue of speeding up *ab initio* MD simulations and finding systems that are well suited for the model. However, due to the small speed-up and increase in performance when compared to VelVer, it does not seem likely that the payoffs of this research are high enough to justify investing resources into it. This is underlined by the fact, that even in the best-performing system investigated in this thesis VelVer performed almost as well as the model. Because of this, it seems unlikely that any adjustments to the model can be made or types of chemical systems can be found where the model greatly outperforms VelVer. If one therefore requires speeding up an *ab initio* MD simulation it would be better to just increase the time step used instead of optimizing and implementing a model that may or may not result in a more accurate simulation.

5 Predicting Forces with a Recurrent Neural Network

As mentioned in subsection 3.3 a way to add additional information about a system to a model is by using molecular descriptors. Molecular descriptors are created through various mathematical and logical steps, which boil an aspect of the chemical system down to a number. Due to the complex nature of chemical systems, there exist many different molecular descriptors each describing a different aspect of the system. Chemical descriptors can be used to get a better understanding of a system, like a molecule's energy or dipole moment, or it can be a more abstract quantity that can be used by models. This means that chemical descriptors have a wide range of applications and they are often used in chemical ML models. However, there is not a direct relationship between molecular descriptors and the forces of the system. This means that regression models, like the VAR_mol model previously used in this thesis, will not be able to use the information contained in chemical descriptors. If one wants to utilize this information to predict forces another type of model is needed.[44,46,47]

Instead of using the statistical VAR models to forecast time series one can use neural networks instead. The most commonly used neural network to predict the next value in a sequence is a recurrent neural network (RNN). An RNN differs from other neural networks by having "memory" so it can retain information from previous predictions. The structure of RNNs can be altered in various ways, and they can even be combined with regular neural network nodes. The exact structure of an RNN can therefore be optimized to the given problem which results in many possible different RNN models. A simple and commonly used RNN model is a standard RNN. It is a model consisting of a single-layer RNN that feeds its output into itself for use in predicting the next point. A standard RNN can therefore be thought of as working like a VAR model, where each predicted value is used to predict the next. However, a problem RNNs can run into is a vanishing gradient when training, which makes it difficult to update the weights of the RNN. The most common way of solving this is by using more complex nodes like LSTM and GRU which contain countermeasures to the vanishing gradient problem. Both the main advantages and disadvantages of an RNN over a VAR come from the fact that an RNN is more complex. This results in it being able to make connections that a VAR model cannot and that an RNN can use more complex data as input than a VAR model can. However, the increased complexity of RNN also means that it is more difficult to train. This results in an increase in the required training data and training time. In addition, the increased complexity of RNN also means an increase in hyperparameters. This makes it more difficult for a user to create a well-working RNN model compared to a VAR model.[45,48]

The ability to handle complex data makes RNN a better choice than the VAR_mol model for utilizing the information present in chemical descriptors. In this section, I will create a standard RNN model consisting of a single GRU node, that uses force trajectories and chemical descriptors to forecast the forces of ethanol. The model will be evaluated based on its MAE and compared to the performance of the VAR_mol model.

5.1 Creating a Recurrent Neural Network

As mentioned a standard RNN model was created. The chosen node for the system was GRU as it generally performs as well as LSTM for numbers while being more memory efficient. Pytorch was used to create the RNN.[49]

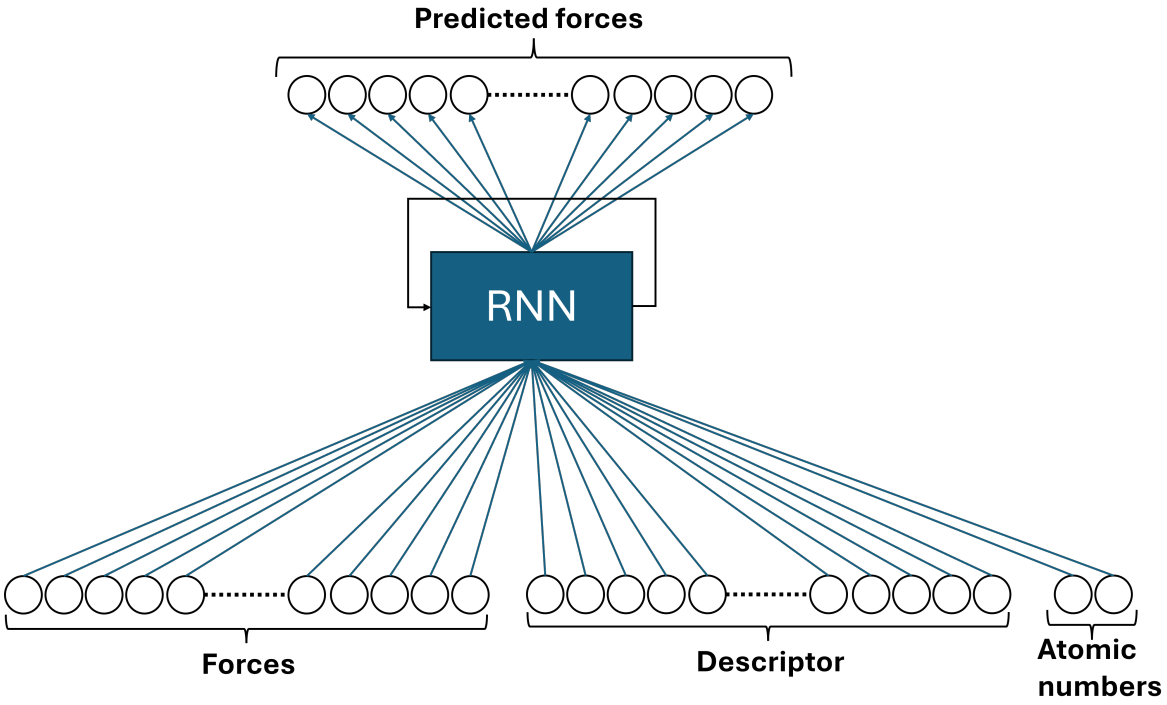


Figure 18: The overall structure of the RNN model is shown. The force trajectories and molecular descriptors for the previous time steps, and the atomic numbers for each atom in the system are used as input to the RNN model. The model then outputs the forces for the next time steps

5.1.1 Finding inputs and outputs

One of the most important steps in creating a good ML model is choosing the input data. As the RNN model is supposed to build upon using force trajectories to predict future forces the past forces of a trajectory will be one of the primary input types. In addition, a molecular descriptor will also be used to give the RNN model information on the chemical system. The molecular descriptor used in this thesis is atom-centered symmetry functions (ACSF) from the library DScribe.[50,51] ACSF uses a fingerprint to describe the local environment of the atoms in the system. It was primarily chosen as it described the environment of the system while still having a relatively low dimensionality. ACSF was the only molecular descriptor tested, and it is therefore possible that other molecular descriptors could be better inputs for the model. The last type of input used by the RNN model is the atomic numbers of the system. Adding identifiers of the system, like atomic numbers, can often help ML models to easier make connections and, in this case, identify and understand any differences between different force trajectories. The RNN model was designed to output the predicted forces of a single time step. If one wants the model to predict more than one time step the output can be used as input just like with a VAR_mol model. A schematic showing the overall structure of the RNN model can be seen in Figure 18.

Having now decided on the types of input in the RNN the next step is figuring out how many time steps in the past should be used. For VAR_mol this was easy to do with a grid search due to the low amount of hyperparameters. However, for RNN this is not feasible due to the large amount of hyperparameters. Another way of judging how many

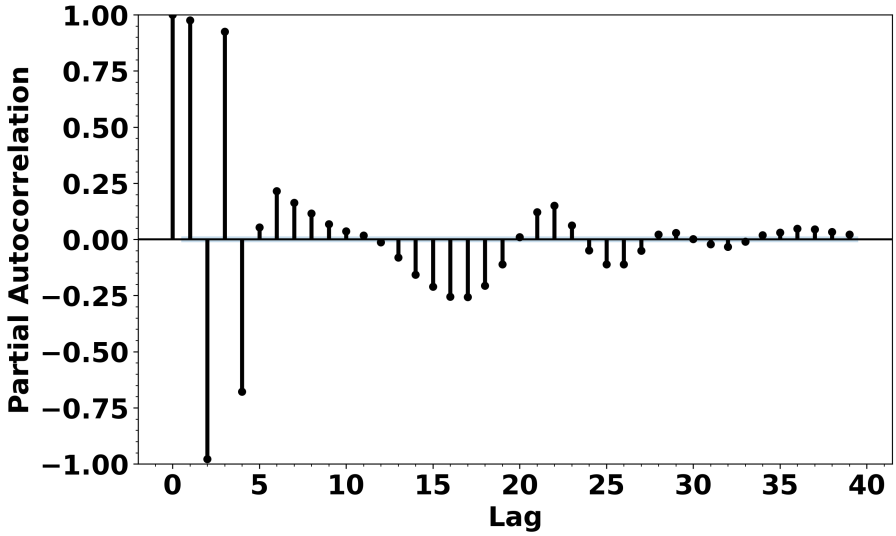


Figure 19: The figure shows the partial autocorrelation of the force trajectory in the x direction for the oxygen atom in ethanol. It shows the autocorrelation of each lagged point taking the information from previous lags into account. It can be seen that most of the information on the trajectory is within the first 4 points in the past as lag = 0 corresponds to a point's correlation with itself.

points in the past contain useful information is therefore needed. This can be done by using partial autocorrelation. It shows the correlation between a time series and a lagged copy of itself while controlling for all previous lags. This means that for $lag = 2$ the partial autocorrelation shows the correlation between x_t and x_{t-2} that is not covered by the correlation between x_t and x_{t-1}. The partial autocorrelation can therefore be used to examine how much information each lag contributes to the present value.[52] Figure 19 shows the partial autocorrelation of oxygen in ethanol's force trajectory in the x direction for 40 lags. Here it can be seen that the partial autocorrelation greatly decreases after $lag = 4$. This value makes sense as most of the well-performing VAR_mol models in the grid search had orders around 4. The inputs for the RNN will therefore be the last 4 time steps of the trajectory.

If there are two inputs, with one input having values in the range of 1000-2000 and the other input having values in the range of 10-20, the first input is likely to dominate the output of the model. If the model does change the weights to give the inputs an equal impact on the output, the model is likely to require more training data and time than if the inputs were of similar size. It can therefore often help the performance of ML models to scale and normalize the input and outputs. The force trajectory and molecular descriptor values were scaled using sklearn's MinMaxScaler, which scales the inputs so that they are between 0 and 1.[53] The atomic numbers were manually scaled using the following formula

$$m_{scaled} = \frac{m_i}{m_{max}} \tag{42}$$

where m_{scaled} is the scaled atomic number, m_i is the atomic number of an atom in the system, and m_{max} is the atomic number of the largest atom in the system. The output forces of the model are also scaled, which means that the scaling needs to be reversed before they can be used to calculate the new momenta and positions of the chemical system.

5.1.2 General and specific trained RNN

The RNN can be trained in two different ways. Firstly, it can be trained so it can be applied at any point in an *ab initio* MD simulation. This can be done by training the model on a whole *ab initio* MD trajectory with many time steps. This type of RNN model will therefore be called RNN_traj. Doing this has the advantage of only having to train the model once after which it can be used to predict the forces at any time step. The large amount of training data also increases the possibility of the RNN_traj model learning any complex connections between molecular descriptors and forces. The main disadvantages of this way of training an RNN_traj model are the large amounts of training data and training time. In order for the model to be applied at any point in a simulation it must be trained to handle all possible scenarios that can be encountered. This is especially bad in chaotic systems where the correlation between scenarios at different points in time is small. This means that the RNN_traj requires a large amount of training data which in practice means many *ab initio* MD time steps calculated with DFT, which takes a long time. This is not such a big problem for common chemical systems, as once trained the model can be used in an infinite number of simulations on the same system. However, it hurts the model's ability to be generally applied to any system, as it requires a large amount of data on a specific system before it can be used.

The other way the RNN can be trained is by training it to a specific section of the trajectory in the same way the VAR_mol model is. As the RNN model will be trained on a window of the trajectory it will be called RNN_window. This model comes with the advantage that much less training data is required as it only needs to be able to predict one scenario. As the information gained from recent time steps is much more relevant to the prediction than information from time steps further away, the model will still have most of the required information to accurately describe the scenario. Because the RNN is trained to a specific section it needs to be retrained each time it is applied in a simulation. However, as the training time is also greatly reduced because of the small amount of training data required, this is not a big problem. The small amount of training data could hinder the model in making complex connections between molecular descriptors and the forces of the molecule.

5.1.3 Training the RNN

The Adam optimizer from Pytorch was used to train both the RNN_traj and RNN_window model.[54] The loss function used for the optimizer is the squared error. This was chosen instead of the absolute error to increase the punishment for the model making large errors, as these have a larger chance of irrevocably altering the simulation. When training a model the Adam optimizer is run for a chosen number of epochs. After these epochs, it is not certain that the model has relaxed to a minimum. In order to ensure that the model has relaxed the Adam optimizer is kept running after reaching the desired number of epochs. This is done until the standard deviation of the loss function over the last 10 epochs is lower than a chosen threshold. After some trial and error a learning rate of $5 \cdot 10^{-3}$ was chosen. A rough grid search was performed to determine the best number of hidden nodes in the GRU node. For RNN_traj 350 was used and for RNN_window 450 was used. RNN models with multiple layers of GRU nodes were also tested but they did not perform better than a RNN model only consisting of a single GRU node. RNN_traj and RNN_window therefore only consist of one GRU node to keep them as simple as possible.

5.2 Evaluating the Recurrent Neural Network

Both an RNN_traj and an RNN_window model were evaluated by calculating the MAE of the models' predictions compared to the forces calculated by DFT. The RNN_traj model was trained on the first 8829 time steps of an *ab initio* MD trajectory for 2000 epochs after which the optimizer was kept running to allow for the loss function to relax. The training for the RNN_traj model took around 30 minutes using an Intel i5-12400F CPU. The RNN_traj model was tested on 2944 time steps. The time steps that are used to test the RNN_traj model are from the same trajectory as the training time steps, but they are from later in the simulation and there is no overlap between the two samples. This replicates how the RNN_traj model would be trained if it were to be applied to a system with no prior data. In this case, the simulation will first be run to gather data to train the model. After enough training data has been acquired the RNN_traj model can be trained and used to predict forces in the rest of the simulation.

The RNN_window model was trained on 9 time steps after which it was used to predict the next forces at the next time step. This number of time steps used in training was chosen based on trial and error where it performed better than using slightly more or less time steps in the training. RNN_window was trained for 100 epochs after which the training kept going until the loss function had relaxed. The training of the RNN_traj model takes around 1 second using an Intel i5-12400F CPU. In order to evaluate the RNN_window model's MAE it was trained and tested at 300 different points in a trajectory all of which had no overlap with each other.

The MAE of the two RNN models and the VAR_mol model can be seen in Figure 20. The RNN_traj model has a slightly lower MAE than the RNN_window model. The increased requirement for training data and training time therefore does come with a slight increase in performance. Based on this, there are underlying patterns in either the force trajectories or the molecular descriptor that the RNN_traj model can utilize which the RNN_window cannot. However, as the difference between the two models is so small it is apparent that the most relevant information for predicting the forces is contained in the recent time steps. This is likely because of the chaotic nature of chemical systems, which means that there is almost no correlation between time steps that are far apart. This results in the RNN_window model fitted to a few recent time steps performing almost as well as the RNN_traj model that is trained on almost 10000 timesteps. While the RNN_traj model also has the advantage of being able to be applied at any point in a simulation, this advantage becomes negligible when considering the difference in training time. Just to make up for taking longer to train the RNN_traj model will have to be applied roughly 1800 times. In addition to this, the RNN_window model can be applied right from the start in a simulation while the RNN_traj model requires a set of training data before it can be applied. The advantage of the RNN_traj model's general applicability therefore mainly comes into play if multiple simulation of the same system is done.

However, when the RNN models are compared to the VAR_mol model it is obvious that they both perform much worse. The VAR_mol model therefore seems to reach the information limit of force trajectories as using more complex models does not increase the performance, even when extra information, in the form of molecular descriptors, is added. It is possible that using more and better training data could improve performance. Instead of just using a whole section of a simulation manually curated time steps that describe different configurations and scenarios of a system could be used. This will make it

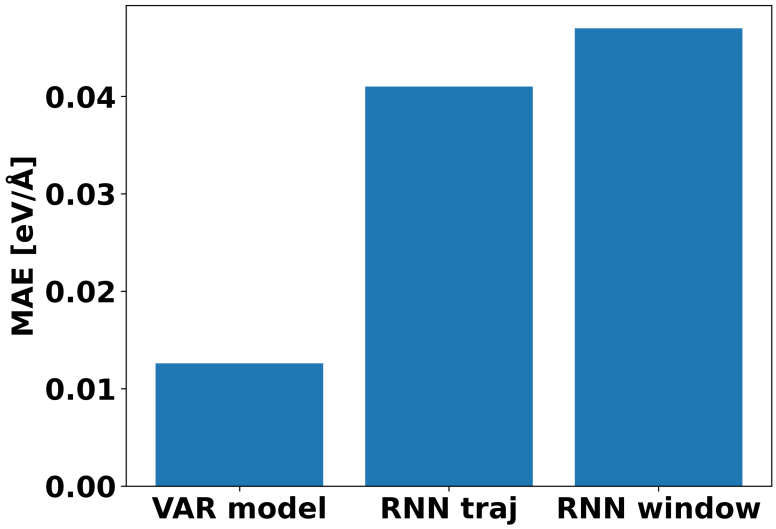


Figure 20: The MAE of the different models are shown. The VAR_mol model is by far the best-performing model. The RNN_traj model performs slightly better than the RNN_window model. However, it requires a large amount of pre-acquired data and takes about 30 minutes to train. The RNN_window requires the same inputs as the VAR_mol model and only takes a second to train. On the other hand. a trained RNN_traj model can be used to predict forces at any point in a simulation, while the RNN_window can only be used once to predict the forces directly after the input data.

so the training data is more efficient at describing the whole phase space of the system. In addition to this choosing better molecular descriptors could also improve the performance of the RNN models. However, doing this requires a large amount of work which will have to be done each time the model needs to forecast a new system. Instead of working so much on an RNN_traj model that at best will still result in a relatively small speed-up, it would be more prudent to focus work on creating ML force fields using molecular descriptors. In this way, the work will at least result in a substantial speed-up with no loss in the general applicability as the RNN_traj models have the same limitations in that regard as ML force fields.

6 Conclusion

In this thesis, a VAR model called VAR_mol was made that could forecast the forces in a system based on its force trajectories. A VAR_mol model fit using 12 data points and using the two most recent points in the past can forecast the forces of ethanol with an MAE of $0.012\,\mathrm{eV\AA^{-1}}$ when forecasting a time step. This MAE is lower than what was achieved by the ML force field created by Chmiela *et al.*[22] However, as their model is a force field it can calculate the forces at all time steps while my model only forecasts one 1 in 13 time steps. This means that their model results in a much larger speed-up than my model and because of this larger errors introduced by it can be tolerated. The MAE of my model is therefore not enough alone to conclude that the model performs well enough to be used in *ab initio* MD simulations.

It is difficult to evaluate whether a model is good enough to be used in an *ab initio* MD simulation. The chaotic nature of chemical systems means that small differences in the

positions of atoms can result in wildly different simulations. These small differences can both come from the initial parameters of the system and differences between the forces predicted by the VAR_mol model and calculated by DFT. For the model to yield a valid solution to an *ab initio* MD simulation the differences between a simulation applying the model and a pure simulation with the same starting parameters must be smaller, than the difference between two pure simulations with different starting parameters. In this thesis, different starting parameters were achieved by using two different seeds to randomly set the initial momenta of the atoms. It is also important that the model does not introduce more errors than if a similar speed-up can be achieved in a more simple way. To test this, the simulations were sped up by increasing the time step used in the VelVer scheme. If the model is not more accurate than achieving the same speed-up with VelVer there is no reason to use it, as it is simpler and easier to just increase the time step.

It is also important that the correct metrics are used when evaluating a model. Many metrics do not give an accurate picture of the overall geometry of the system. This means that even if the model performs well on those metrics it is not a guarantee that the geometry and properties of the system are kept intact by the model. This was encountered when evaluating my VAR_mol model's performance on ethanol using average values and distributions of angles and bond lengths, RMSD of the system, and the distribution of conformational isomers. According to all of these metrics, my model yielded valid solutions to the *ab initio* MD simulation. The metrics also showed that although VelVer was better at describing the main carbon chain and overall movements of ethanol, my model was better at describing the hydroxyl group. However, this was not the case when examining 2D histograms of dihedral angles in ethanol, which gives better insight into the overall geometry of the systems than the previous methods. In the case of applying my model to ethanol, 2D histograms of dihedral angles showed that the model did not yield a valid solution. In addition to this, speeding up the simulations using VelVer was more accurate than using my model. Because of metrics analyzing different aspects of chemical systems, it is important to be conscious of the metrics used, either by yourself or others, to evaluate a model. Luckily 2D plots of dihedral angles are the most widespread metric and are especially often used to evaluate ML force fields.

One reason for my VAR_mol model performing badly on ethanol could be the small size of the molecule. As the model is applied many times over a simulation at some point an unlikely large error will occur. As the ethanol system only has 27 force trajectories a large error in one force trajectory will have a large impact. Because the system is chaotic any small differences in positions matter a lot. The impact of a single large error will therefore be propagated throughout the whole simulation, resulting in it being completely different from a pure simulation without the model. A way to reduce the effect of unlikely large errors is to apply the model to systems with more atoms. This will create a buffer of sorts that makes sure that there are many force trajectories with small errors that can counteract the large errors introduced once in a while. In order to test this the model was applied to aspirin and cholesterol. The model's performance on these systems was evaluated using 2D histograms of dihedral angles. This showed that the model yielded valid solutions for both aspirin and cholesterol, which is a stark improvement to the model's performance on ethanol. When comparing the model to achieving the same speed-up with VelVer, the model performed slightly better than VelVer on aspirin and slightly worse on cholesterol.

In this thesis, I also explored whether RNN could be a better alternative to the VAR_mol models for using force trajectories to forecast forces. As RNN models are

more complex than VAR_mol models it can make use of more complex information. This means that it can use chemical descriptors that describe the environment of the chemical system in addition to the force trajectories. While this extra information and more complex model sounds nice in theory, in reality, my VAR_mol model performed much better than all tested RNN models. It is possible that the performance of the RNN models can be improved. This could be done by either using better and more relevant molecular descriptors or by improving the phase space described by the training data. However, this will require a lot of work for a small speed-up. If one does want to work on creating good training data and describing the chemical systems using molecular descriptors it would be better to apply it in creating ML force fields, as the payoff for the work will then be much higher.

In conclusion, my VAR_mol model was not able to describe ethanol adequately. However, the model performed much better on relatively heavy molecules than on ethanol. The model performed slightly better than increasing the time step for speeding up simulations of aspirin. This means that there do exist systems for which using my VAR_mol model to speed up the simulation is better than doing it by increasing the time step used in VelVer. However, the small increase in performance gained by using my model is most likely not worth the hassle of implementing it and making sure that the system of interest is well-suited to the model. In addition to this, RNN models based mainly on force trajectories also performed badly. It, therefore, seems that force trajectories do not contain enough information to be used to forecast forces accurately enough to be used in *ab initio* MD simulation. For almost all systems it would be better to speed up the simulations by just increasing the time step.

Acknowledgements

I want to thank Gemma C. Solomon and William Bro-Jørgensen for guiding me through my thesis, and Tomislav Rožić and Matthew S. Teynor for enlightening discussions of my work.

References

[1] Ben Leimkuhler and Charles Matthews. Molecular dynamics. *Interdisciplinary applied mathematics*, 39(1), 2015.

[2] Scott A. Hollingsworth and Ron O. Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018.

[3] Dominik Marx and Jürg Hutter. *Ab Initio Molecular Dynamics: Basic Theory and Advanced Methods*. Cambridge University Press, Cambridge, 2009.

[4] Godehard Sutmann. Classical molecular dynamics, 2002.

[5] M. A. González. Force fields and molecular dynamics simulations. *École thématique de la Société Française de la Neutronique*, 12:169–200, 2011.

[6] Itai Leven, Hongxia Hao, Songchen Tan, Xingyi Guan, Katheryn A. Penrod, Dooman Akbarian, Benjamin Evangelisti, Md Jamil Hossain, Md Mahbubul Islam, Jason P.

Koski, Stan Moore, Hasan Metin Aktulga, Adri C. T. van Duin, and Teresa Head-Gordon. Recent advances for improving the accuracy, transferability, and efficiency of reactive force fields. *Journal of Chemical Theory and Computation*, 17(6):3237–3251, 2021. doi: 10.1021/acs.jctc.1c00118.

[7] Adri C. T. van Duin, Siddharth Dasgupta, Francois Lorant, and William A. Goddard. Reaxff: A reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, 105(41):9396–9409, 2001. doi: 10.1021/jp004368u.

[8] Thomas P. Senftle, Sungwook Hong, Md Mahbubul Islam, Sudhir B. Kylasa, Yuanxia Zheng, Yun Kyung Shin, Chad Junkermeier, Roman Engel-Herbert, Michael J. Janik, Hasan Metin Aktulga, Toon Verstraelen, Ananth Grama, and Adri C. T. van Duin. The reaxff reactive force-field: development, applications and future directions. *npj Computational Materials*, 2(1):15011, 2016.

[9] Nathan Argaman and Guy Makov. Density functional theory: An introduction. *American Journal of Physics*, 68(1):69–79, 2000.

[10] Jacob D. Durrant and J. Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC Biology*, 9(1):71, 2011.

[11] Marco De Vivo, Matteo Masetti, Giovanni Bottegoni, and Andrea Cavalli. Role of molecular dynamics and related methods in drug discovery. *Journal of Medicinal Chemistry*, 59(9):4035–4061, 2016. doi: 10.1021/acs.jmedchem.5b01684.

[12] Paolo Carloni, Ursula Rothlisberger, and Michele Parrinello. The role and perspective of ab initio molecular dynamics in the study of biological systems. *Accounts of Chemical Research*, 35(6):455–464, 2002. doi: 10.1021/ar010018u.

[13] Jamshed Anwar and Dirk Zahn. Uncovering molecular processes in crystal nucleation and growth by using molecular simulation. *Angewandte Chemie International Edition*, 50(9):1996–2013, 2011.

[14] Danny Perez, Blas P. Uberuaga, Yunsic Shim, Jacques G. Amar, and Arthur F. Voter. *Chapter 4 Accelerated Molecular Dynamics Methods: Introduction and Recent Developments*, volume 5, pages 79–98. Elsevier, 2009.

[15] Radu A. Miron and Kristen A. Fichthorn. Accelerated molecular dynamics with the bond-boost method. *The Journal of Chemical Physics*, 119(12):6210–6216, 2003.

[16] Blas P. Uberuaga, Francesco Montalenti, Timothy C. Germann, and Arthur F. Voter. *Accelerated Molecular Dynamics Methods*, pages 629–648. Springer Netherlands, Dordrecht, 2005.

[17] Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. Machine learning force fields. *Chemical Reviews*, 121(16):10142–10186, 2021. doi: 10.1021/acs.chemrev.0c01111.

[18] Tran Doan Huan, Rohit Batra, James Chapman, Sridevi Krishnan, Lihua Chen, and Rampi Ramprasad. A universal strategy for the creation of machine learning-based atomistic force fields. *npj Computational Materials*, 3(1):37, 2017.

[19] B. Brutovsky, T. Mülders, and G. R. Kneller. Accelerating molecular dynamics simulations by linear prediction of time series. *The Journal of Chemical Physics*, 118(14):6179–6187, 2003.

[20] Branislav Brutovsky and Gerald R. Kneller. Linear prediction of force time series to accelerate molecular dynamics simulations. *Computer Physics Communications*, 169(1):339–342, 2005.

[21] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis*, book section Chapter one: Introduction, pages 1–6. Wiley, 4. edition, 2008.

[22] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[23] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

[24] Graham Elliott and Allan Timmermann. Forecasting in economics and finance. *Annual Review of Economics*, 8(Volume 8, 2016):81–110, 2016.

[25] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[26] G. Mahalakshmi, S. Sridevi, and S. Rajaram. A survey on forecasting of time series data. IEEE, 2016.

[27] Mehmet Tektaş. Weather forecasting using anfis and arima models. *Environmental Research, Engineering and Management*, 51(1):5–10, 2010.

[28] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis*, book section Chapter 1.2 Stochastic and Deterministic Dynamic Mathematical Models, pages 7–10. Wiley, 4. edition, 2008.

[29] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis*, book section Chapter 14: Multivariate Time Series Analysis, pages 551–561. Wiley, 4. edition, 2008.

[30] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N. Groves, Bjørk Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W. Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[31] Jens Jørgen Mortensen, Ask Hjorth Larsen, Mikael Kuisma, Aleksei V. Ivanov, Alireza Taghizadeh, Andrew Peterson, Anubhab Haldar, Asmus Ougaard Dohn, Christian Schäfer, Elvar Örn Jónsson, Eric D. Hermes, Fredrik Andreas Nilsson, Georg Kastlunger, Gianluca Levi, Hannes Jónsson, Hannu Häkkinen, Jakub Fojt, Jiban Kangsabanik, Joachim Sødequist, Jouko Lehtomäki, Julian Heske, Jussi Enkovaara, Kirsten Trøstrup Winther, Marcin Dulak, Marko M. Melander, Martin Ovesen, Martti Louhivuori, Michael Walter, Morten Gjerding, Olga Lopez-Acevedo, Paul Erhart, Robert Warmbier, Rolf Würdemann, Sami Kaappa, Simone Latini, Tara Maria Boland, Thomas Bligaard, Thorbjørn Skovhus, Toma Susi, Tristan Maxson, Tuomas Rossi, Xi Chen, Yorick Leonard A. Schmerwitz, Jakob Schiøtz, Thomas Olsen, Karsten Wedel Jacobsen, and Kristian Sommer Thygesen. Gpaw: An open python package for electronic structure calculations. *The Journal of Chemical Physics*, 160(9):092503, 2024.

[32] D. Tufts and R. Kumaresan. Singular value decomposition and improved frequency estimation using linear prediction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(4):671–675, 1982.

[33] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, book section Chapter 1: Singular Value Decomposition (SVD), pages 3–46. Cambridge University Press, Cambridge, 2019.

[34] Peter J. Bickel, Bo Li, Alexandre B. Tsybakov, Sara A. Van De Geer, Bin Yu, Teófilo Valdés, Carlos Rivero, Jianqing Fan, and Aad Van Der Vaart. Regularization in statistics. *Test*, 15(2):271–344, 2006.

[35] Richard C. Aster, Brian Borchers, and Clifford H. Thurber. *Parameter Estimation and Inverse Problems*, volume 90, book section 4 Tikhonov regularization, pages 89–118. Academic Press, 2005.

[36] I. P. Omelyan, I. M. Mryglod, and R. Folk. Optimized verlet-like algorithms for molecular dynamics simulations. *Physical Review E*, 65(5), 2002.

[37] R. Han and S. Luber. Trajectory-based machine learning method and its application to molecular dynamics. *Molecular Physics*, 118(19-20):e1788189, 2020.

[38] Vance W. Berger and YanYan Zhou. *Kolmogorov–Smirnov Test: Overview*. 2014. Major Reference Works.

[39] John I. Marden. Positions and qq plots. *Statistical Science*, 19(4):606–614, 2004.

[40] W. Humphrey, A. Dalke, and K. Schulten. Vmd: visual molecular dynamics. *J Mol Graph*, 14(1):33–8, 27–8, 1996. Humphrey, W Dalke, A Schulten, K 5 P41 RR05969-04/RR/NCRR NIH HHS/United States Journal Article Research Support, Non-U.S. Gov't Research Support, U.S. Gov't, Non-P.H.S. Research Support, U.S. Gov't, P.H.S. United States 1996/02/01 J Mol Graph. 1996 Feb;14(1):33-8, 27-8. doi: 10.1016/0263-7855(96)00018-5.

[41] Steve Scheiner and Paul G. Seybold. Quantum chemical analysis of the energetics of the anti and gauche conformers of ethanol. *Structural Chemistry*, 20(1):43–48, 2009.

[42] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E. Charron, Gianni De Fabritiis, Frank Noé, and Cecilia Clementi. Machine learning of coarse-grained molecular dynamics force fields. *ACS Central Science*, 5(5):755–767, 2019.

[43] Raimondas Galvelis, Alejandro Varela-Rial, Stefan Doerr, Roberto Fino, Peter Eastman, Thomas E. Markland, John D. Chodera, and Gianni De Fabritiis. Nnp/mm: Accelerating molecular dynamics simulations with machine learning potentials and molecular mechanics. *Journal of Chemical Information and Modeling*, 63(18):5701–5708, 2023. doi: 10.1021/acs.jcim.3c00773.

[44] Roberto Todeschini and Viviana Consonni. *Handbook of molecular descriptors*. John Wiley & Sons, 2008.

[45] Larry R. Medsker and Lakhmi Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[46] L. Xue and J. Bajorath. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Combinatorial Chemistry & High Throughput Screening*, 3(5):363–372, 2000.

[47] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[48] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*, page Article 721. Curran Associates Inc., 2019.

[50] Lauri Himanen, Marc O. J. Jäger, Eiaki V. Morooka, Filippo Federici Canova, Yashasvi S. Ranawat, David Z. Gao, Patrick Rinke, and Adam S. Foster. Dscribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, 247:106949, 2020.

[51] Jarno Laakso, Lauri Himanen, Henrietta Homm, Eiaki V. Morooka, Marc O. J. Jäger, Milica Todorović, and Patrick Rinke. Updates to the dscribe library: New descriptors and derivatives. *The Journal of Chemical Physics*, 158(23):234802, 2023.

[52] J. H. F. Flores, P. M. Engel, and R. C. Pinto. Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2012.

[53] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher,

Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, 2011.

[54] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

A VAR Model Code

```
import numpy as np

class ShavedVAR:
"""
Fit VAR(p) process and do lag order selection

Parameters
----------
endog : array_like
    2-d endogenous response variable. The independent variable.
"""

def __init__(self, endog):
    self.endog = endog
    self.neqs = self.endog.shape[1]

def fit(
    self,
    maxlags: int,
    alpha: int,
    method = 'SVD'
):
    """
    Fit the VAR model

    Parameters
    ----------
    maxlags : {int, None}, default None
        Maximum number of lags to check for order selection,
           defaults to
        12 * (nobs/100.)**(1./4), see select_order function
    alpha : float
        L2 regularization constant
    method : string
        Which method is used to find parameters, SVD is default

    Returns
    -------
    VARResults
        Estimation results
    """
    lags = maxlags
    return self._estimate_var(lags, alpha, method=method)

def _estimate_var(self, lags, alpha, method):
    """
    lags : int
        Lags of the endogenous variable.
    alpha : float
```

```
49          L2 regularization constant
50      method : string
51          Which method is used to find parameters, SVD is default
52      """
53      endog = self.endog
54      nobs = len(endog)
55
56      # Ravel C order, need to put in descending order
57      lb_minus_lags = nobs - lags
58      z = np.zeros((lb_minus_lags, endog.shape[1] * lags))
59      for t in range(lags, nobs):
60          t_minus_lags = t - lags
61          middle_cols = endog[t_minus_lags:t, :][::-1, :].ravel()
62          z[t_minus_lags, :] = middle_cols
63
64      y_sample = endog[lags:]
65      if method=="SVD":
66
67          U, s, Vt = np.linalg.svd(z, full_matrices=False)
68          # Create Sigma matrix from 1D array
69          sigma = np.diag(s)
70          # Create a Sigma squared matrix
71          sigma_sq = np.dot(sigma, sigma)
72          # Apply penalty to the singular values
73          sigma_a = sigma_sq + np.identity(sigma.shape[0]) * alpha
74          # Calculate inverse
75          sigma_a_inv = np.diag(1/np.diag(sigma_a))
76          # Calculate params
77          UTy = np.dot(U.T, y_sample)
78          Sigma_UTy = np.dot(sigma, UTy)
79          sigma_penilized = np.dot(sigma_a_inv, Sigma_UTy)
80
81          # Calculate coefficients
82          params = np.dot(Vt.T, sigma_penilized)
83      elif method=="Inverse":
84          params = np.dot(np.linalg.inv(np.dot(z.T, z)+alpha*np.
               identity(lags*endog.shape[1])), np.dot(z.T, y_sample))
85      else:
86          print("ERROR: invalid method chosen. Please choose 'SVD'
               or 'Inverse' as the method")
87
88
89      return VARProcess(params, lags)
```

B Integration of Model Code

```
"""
Integrates the force predictions from the model with the Velocity
   Verlet algorithm from ASE

Parameters
------------
mol : ASE.atoms
    contains the information on the chemical system
n_time_steps : int
    Number of time steps to run the simulation for
input : int
    The input, t, used to fit the model
pred_step : int
    Number of predictions made by the model
predict_forces : function
    Fits a model and uses it to predict the upcoming forces
"""

for i in range(n_time_steps):
    if i%(input-1) == 0 and i!=0:
        predicted_forces = predict_forces()
        for j in range(pred_step+1):
            #Get masses for the atoms in the molecule
            masses = mol.get_masses()[:, np.newaxis]

            #Get the forces, momenta, and positions for the
               current step
            if j == 0:
                forces = mol.get_forces()
            else:
                forces = predicted_forces[j-1,:]
                forces = forces.reshape(mol.
                   get_global_number_of_atoms(), 3)

            p = mol.get_momenta()
            r = mol.get_positions()

            #Calculate new momenta and positions
            p += 0.5 * dt * forces
            mol.set_positions(r + dt * p / masses)

            if mol.constraints:
                p = (mol.get_positions() - r) * masses / dt

            #Momenta needs to be stored before possible
               calculations of forces
            mol.set_momenta(p, apply_constraint=False)

            #Forces for next step is found either using predicted
```

```
                forces or gpaw calculator
            if j<pred_step:
                forces = predicted_forces[j,:]
                forces = forces.reshape(mol.
                    get_global_number_of_atoms(), 3)
                print_forces(md=0, forces=forces)
            else:
                forces = mol.get_forces(md=True)
                print_forces(md=1, forces=forces)

            #Calculate and set momenta for the next step
            mol.set_momenta(mol.get_momenta() + 0.5 * dt * forces)
            write(file+".xyz", mol, append = True)

    dyn.run(1)
    print_forces()
    write(file+".xyz", mol, append = True)
    traj.write()
```

C Average Values

Table 6: Average values of angles for pure simulation, models, and VelVer.

Seed	**Simulation**	**H-C-C** [°]	**H-C-H** [°]	**C-C-O** [°]	**H-C-O** [°]	**C-O-H** [°]
1	Pure simulation	110.74	108.14	110.79	112.54	102.57
	$t = 12, p = 2, \lambda = 0$	110.45	107.80	110.31	110.79	102.61
	$t = 15, p = 4, \lambda = 0$	110.00	108.02	109.79	110.14	102.87
	$t = 12, p = 2, \lambda \approx 1.2 \cdot 10^{-4}$	110.45	107.88	111.17	112.03	102.76
	$t = 15, p = 4, \lambda \approx 8.6 \cdot 10^{-5}$	110.10	107.75	111.08	111.16	102.66
	VelVer speed-up	110.54	107.81	110.62	111.11	102.52
2	Pure simulation	110.41	107.88	108.35	111.94	102.86
	$t = 12, p = 2, \lambda = 0$	110.42	107.93	109.51	111.10	102.84
	$t = 15, p = 4, \lambda = 0$	110.23	108.07	109.05	112.43	102.93
	$t = 12, p = 2, \lambda \approx 1.2 \cdot 10^{-4}$	110.50	107.90	110.95	110.78	102.63
	$t = 15, p = 4, \lambda \approx 8.6 \cdot 10^{-5}$	110.17	107.79	110.19	111.53	102.64
	VelVer speed-up	110.46	108.11	108.06	111.80	102.96

Table 7: Average values of bond lengths for pure simulation, models, and VelVer.

Seed	**Simulation**	**C-C** [Å]	**C-H** [Å]	**C-O** [Å]	**O-H** [Å]
1	Pure simulation	1.614	1.210	1.539	1.121
	$t = 12, p = 2, \lambda = 0$	1.621	1.211	1.546	1.124
	$t = 15, p = 4, \lambda = 0$	1.619	1.211	1.542	1.129
	$t = 12, p = 2, \lambda = 1.16 \cdot 10^{-4}$	1.624	1.214	1.545	1.117
	$t = 15, p = 4, \lambda = 8.572 \cdot 10^{-5}$	1.627	1.217	1.549	1.127
	VelVer speed-up	1.616	1.208	1.543	1.116
2	Pure simulation	1.605	1.211	1.534	1.125
	$t = 12, p = 2, \lambda = 0$	1.617	1.210	1.542	1.114
	$t = 15, p = 4, \lambda = 0$	1.614	1.217	1.541	1.118
	$t = 12, p = 2, \lambda = 1.16 \cdot 10^{-4}$	1.617	1.210	1.542	1.122
	$t = 15, p = 4, \lambda = 8.572 \cdot 10^{-5}$	1.620	1.215	1.543	1.116
	VelVer speed-up	1.611	1.212	1.536	1.110

D Distributions

D.1 Two Sample KS Test

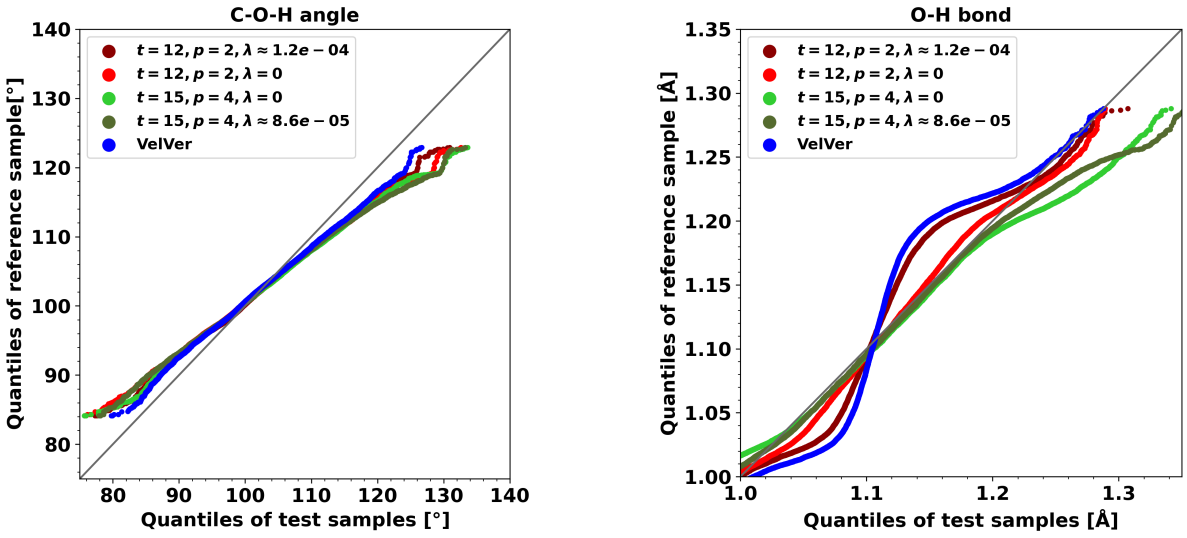
Table 8: P-values from KS test of angles for pure simulation, models, and VelVer.

Seed	Simulation	**H-C-C**	**H-C-H**	**C-C-O**	**H-C-O**	**C-O-H**
1	Pure simulation	1.00	1.00	1.0	1.00	1.0
	$t = 12, p = 2, \lambda = 0$	1.2×10^{-4}	2.0×10^{-14}	5.6×10^{-100}	4.8×10^{-102}	3.7×10^{-45}
	$t = 15, p = 4, \lambda = 0$	6.4×10^{-21}	6.6×10^{-4}	8.3×10^{-305}	5.4×10^{-208}	2.4×10^{-49}
	$t = 12, p = 2, \lambda \approx 10^{-4}$	5.2×10^{-13}	3.1×10^{-26}	1.2×10^{-312}	1.6×10^{-15}	6.1×10^{-34}
	$t = 15, p = 4, \lambda \approx 10^{-4}$	6.3×10^{-25}	1.1×10^{-35}	9.8×10^{-154}	1.3×10^{-76}	2.4×10^{-46}
	VelVer speed-up	6.6×10^{-6}	3.9×10^{-46}	2.1×10^{-69}	2.6×10^{-78}	6.8×10^{-35}
2	Pure simulation	1.2×10^{-36}	7.5×10^{-42}	0.0	5.5×10^{-38}	6.3×10^{-7}
	$t = 12, p = 2, \lambda = 0$	5.5×10^{-7}	9.5×10^{-5}	2.7×10^{-70}	2.3×10^{-54}	4.7×10^{-8}
	$t = 15, p = 4, \lambda = 0$	3.1×10^{-18}	1.6×10^{-9}	1.1×10^{-233}	4.2×10^{-3}	5.2×10^{-28}
	$t = 12, p = 2, \lambda \approx 10^{-4}$	1.4×10^{-14}	2.0×10^{-27}	1.2×10^{-38}	2.9×10^{-69}	1.5×10^{-23}
	$t = 15, p = 4, \lambda \approx 10^{-4}$	2.6×10^{-10}	3.6×10^{-21}	3.2×10^{-177}	4.2×10^{-37}	3.3×10^{-6}
	VelVer speed-up	3.2×10^{-6}	4.2×10^{-10}	0.00	6.9×10^{-23}	1.8×10^{-10}

Table 9: P-values from KS test of bond lengths for pure simulation, models, and VelVer.

Seed	**Simulation**	**C-C**	**C-H**	**C-O**	**O-H**
1	Pure simulation	1	1	1	1
	$t = 12, p = 2, \lambda = 0$	5.3×10^{-29}	2.2×10^{-2}	1.6×10^{-11}	1.9×10^{-45}
	$t = 15, p = 4, \lambda = 0$	4.1×10^{-13}	1.7×10^{-6}	4.9×10^{-4}	5.0×10^{-126}
	$t = 12, p = 2, \lambda = 1.16 \cdot 10^{-4}$	2.4×10^{-33}	1.1×10^{-7}	1.6×10^{-9}	6.1×10^{-277}
	$t = 15, p = 4, \lambda = 8.572 \cdot 10^{-5}$	2.9×10^{-57}	9.6×10^{-16}	9.9×10^{-45}	1.3×10^{-38}
	VelVer speed-up	3.2×10^{-7}	5.3×10^{-8}	1.5×10^{-7}	0
2	Pure simulation	2.4×10^{-36}	2.0×10^{-33}	3.0×10^{-61}	5.7×10^{-80}
	$t = 12, p = 2, \lambda = 0$	1.5×10^{-6}	3.6×10^{-3}	3.3×10^{-12}	2.4×10^{-170}
	$t = 15, p = 4, \lambda = 0$	2.9×10^{-6}	1.3×10^{-62}	1.4×10^{-9}	1.39×10^{-50}
	$t = 12, p = 2, \lambda = 1.16 \cdot 10^{-4}$	2.0×10^{-16}	2.4×10^{-14}	2.5×10^{-11}	2.9×10^{-26}
	$t = 15, p = 4, \lambda = 8.572 \cdot 10^{-5}$	5.6×10^{-9}	2.2×10^{-43}	5.5×10^{-5}	8.4×10^{-183}
	VelVer speed-up	1.6×10^{-21}	1.8×10^{-18}	4.5×10^{-50}	0

D.2 Q-Q plot of tested models



(a) (b)

Figure 21: The figure shows the Q-Q plots of the 4 different models tested and VelVer with a pure simulations used as reference. All the simulatons were done with the same seed. Overall all the models and VelVer performs similarly for the distribution of the C-O-H angle. For the distribution of the O-H bond the $t = 12, p = 2, \lambda = 0$ model performed the best, but all models outperformed VelVer.

E RMSD

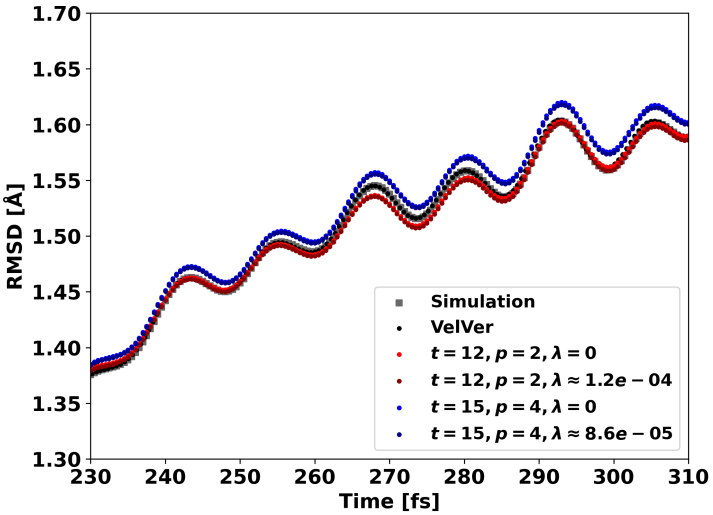


Figure 22: The RMSD of the 4 tested models and VelVer is shown. The RMSD is calculated between the positions of the atoms of the molecule in each frame and the positions in the initial frame of the simulations. The models perform very similarly but the two $t = 12, p = 2$ models seem to perform slightly better.

F Conformational Isomers

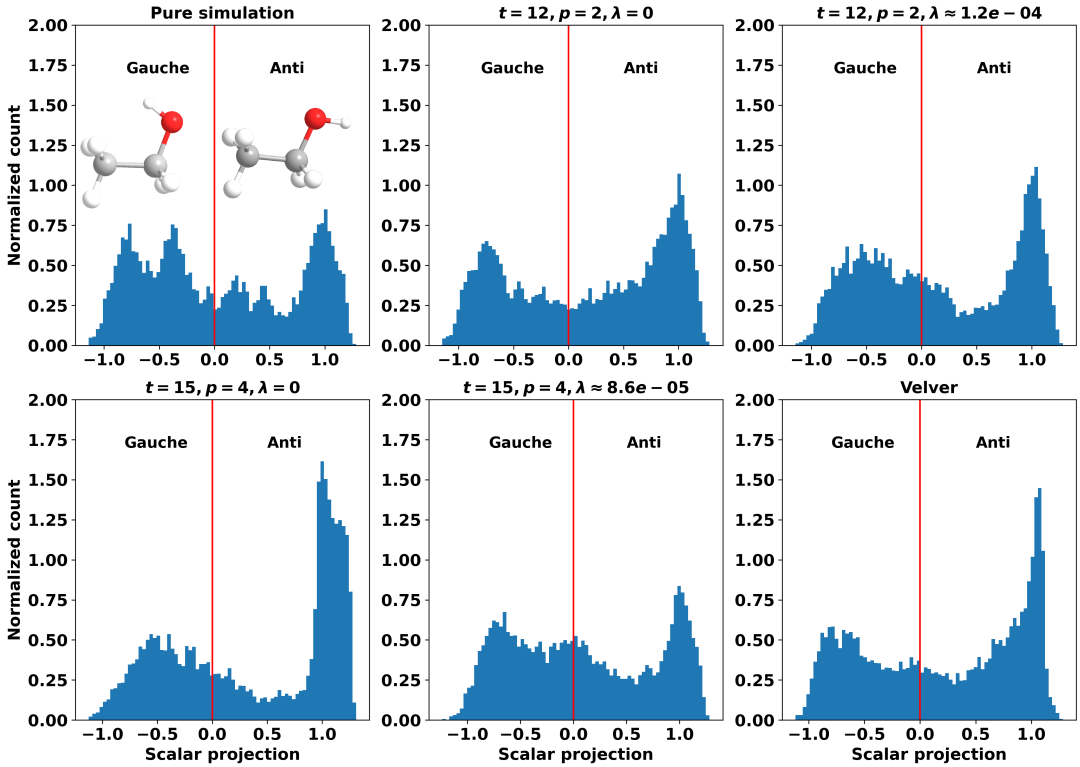


Figure 23: The figure shows the distribution of the scalar vector projection for the different models tested, a pure simulation, and VelVer. All of the simulations were done with the same seed. The two models that replicate the pure simulations most is $t = 12, p = 2, \lambda = 0$ and $t = 15, p = 4, \lambda \approx 8.6 \cdot 10^{-5}$.

G 2D Dihedral

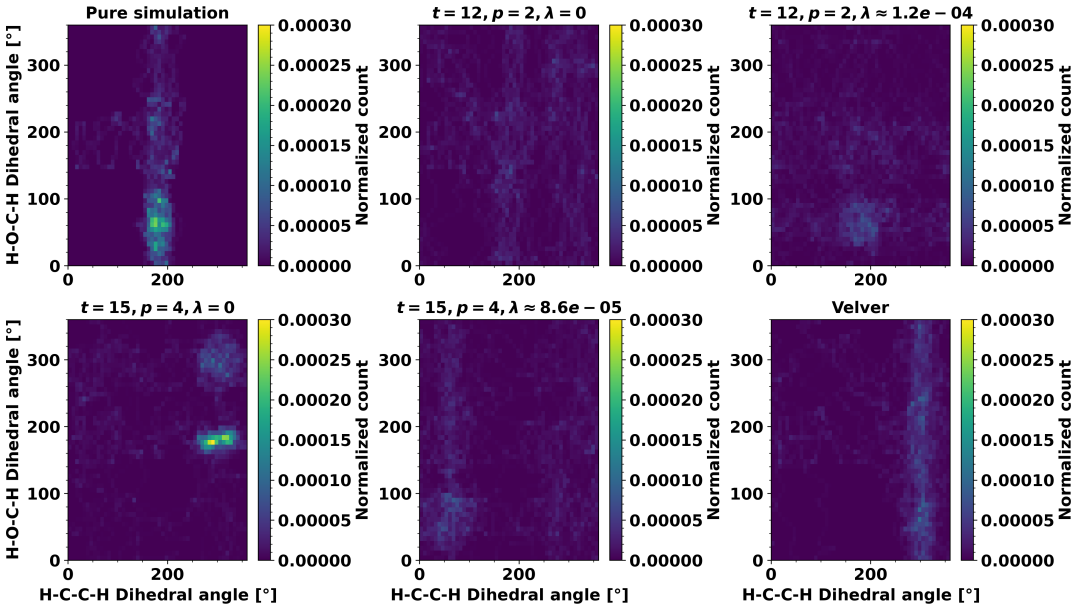


Figure 24: The figure shows the 2D dihedral histograms of the four tested models, a pure simulation, and VelVer all of which were done with the same simulation. The VelVer histogram is closer to the pure simulation than any of the models. The two models that resemble the pure simulation the most are $t = 12, p = 2, \lambda = 0$ and $t = 15, p = 4, \lambda \approx 8.6 \cdot 10^{-5}$.