

## Table of Contents

Table of Contents	1
Generating A Moment Tensor Potential for HfO2 Using Active Learning	2
Background	2
Getting Started	3
Workflow	4
Step 1: Prepare Initial Reference Configurations	5
Step 2: Compute Reference Data and Setup Active Learning	5
Calculator: The reference calculator to use for MTP training	6
Snippet 1: Import reference geometries	6
Snippet 2: Compute reference data	6
Snippet 3: Customize MTP training	6
Snippet 4: Set up active learning	6
Step 3: Find an MTP with Lowest Error	7
Validation MD Simulation	9
References	10
FAQ Section	10
Q1. In step 1, one prepares the initial configurations for their system. How important is it to define all of the different phases of the material since in the next step (during active learning), the missing configurations are added anyways?	10
Q2. Is step 1 always necessary or can one start from active learning directly?	11
Q3. This question is about the fitting parameter options in step 2, snippet3. Is there some way to guess these options for particular system - for example cutoff radii (check if they have any sense) without actually finishing step 2.	11
Q4. Under non-linear parameters optimization, what else could be optimized if energy_only option is False and when this is recommended?	11
Q5. In the user example for ActiveLearningSimulations in the manual, Hf and O isolated energies are given in the example script but here, in the tutorial, we don't have them defined. Why is that?	11
Q6. In step 3, it is said that this may take several hours to get done. Where could one define the parallel options? Would it be possible to do in parallel the steps from the loop?	11
Q7. Regarding the MTP_training.py and MTP_training_best.py, the former contains a loop over random seeds values, while in the second one the value is given. Is the best (lowest RMSE) seed chosen automatically (given somewhere in main output of MTP_training.py) or user need to actually screen all these mtp_fit_mtp_i.log files?	11
Q8. This question is about validation MD simulation step. MD.py script contains MTP obtained in the step 3. But on the last two pictures there displayed also results of DFT-MD calculations. So what basis sets were used for the DFT-MD run?	12


[Try it!](#)[QuantumATK](#)[Contact](#)[Docs](#) » [Tutorials](#) » [New or Recently Updated Tutorials](#) »Generating A Moment Tensor Potential for HfO<sub>2</sub> Using Active Learning

## Generating A Moment Tensor Potential for HfO<sub>2</sub> Using Active Learning

Version: T-2022.03

### Downloads & Links

[PDF version](#)  
[Basic QuantumATK Tutorial](#)  
[ATK Reference Manual](#)

In this tutorial, we will train a Moment Tensor Potential (MTP) for bulk HfO<sub>2</sub> referenced to DFT using QuantumATK .



### Background

Machine learned interatomic potentials (MLIPs) have become popular in recent years since they can provide ab initio level of accuracy in energies, forces and stresses needed for MD simulations, geometry optimizations and single point calculations at modest computational effort. Moment tensor potentials (MTPs) are a class of MLIPs that have been identified to provide high performance for a given accuracy when compared to other MLIP models in the literature <sup>[1]</sup>. On training an MTP for a system of interest, our goal is to establish a structure-property relationship - here, 'structure' is the atomistic description of the system (3N Cartesian coordinates) and 'property' is the system's potential energy surface (PES - 3N dimensional function).

Within MTP framework, the total energy of a configuration is computed as the sum of energy contributions from atomic neighborhoods. Atomic neighborhoods are defined as the representation of the immediate chemical environment of each atom in the system within a cutoff radius. Since Cartesian coordinates are not invariant to translations, rotations and permutations of atoms in the configuration, they can not be used as such as inputs in MTP training. Moment tensors include the radial, angular, and many-body distribution of atoms within a cutoff sphere. Their contractions into scalar basis functions provide an invariant representation of atomic environments ('structure') needed as input for MTP training <sup>[2]</sup>. An electronic structure method, such as density functional theory (DFT), that describes the system of interest accurately is chosen as the 'reference method' to compute the 'properties' of interest. A 'reference data set' includes the reference configurations representing the system of interest along with their energies (E), atomic forces (F) and stress tensors (S) computed using the chosen reference method.

A MTP can be trained by fitting a linear regression model to the reference data set. MTP training on a reference data set can be performed in two ways: batch learning and active learning. In batch learning, an MTP is trained on a precomputed reference data set. In active learning, a preliminary reference data set is improved on-the-fly by actively adding missing configurations to it. This is done by first training a preliminary MTP on the initial reference data and starting an MD simulation using this MTP potential. When geometries that are very different from the initial reference geometries are encountered in the MD, they are added to the reference data set after computing their E, F and S using the reference method as described in [3] and [4].

It is important to understand that MTPs or any other MLIPs are trained to describe a system accurately (very close to the reference method) within the configurational space of the system sampled in the training data set. Unlike many empirical potentials with physical functional forms, MLIPs are not highly transferable. Therefore, an MLIP trained for a bulk material might not describe free surfaces or nanoparticles of that material accurately unless such geometries are explicitly included in the training data set. Moreover, an MLIP trained for room temperature behavior of a system might not work reliably to describe the system at high temperatures unless disordered structures relevant at high temperatures are included in the training dataset. To improve transferability, active learning of the system at its desired state (high temperatures or surface geometries) should be carried out in order to include the missing reference data.

In this tutorial we will generate an MTP for bulk  $\text{HfO}_2$  that can predict DFT quality E, F and S of both crystalline and amorphous configurations using **NanoLab** GUI. Since DFT is used as reference method, this will take 4 to 6 days depending on the computational resources available.

#### Tip


If you are only interested in training an MTP to simulate crystal structures, you can skip the active learning part and go from the initial reference geometries directly to the final training in Step 3.

## Getting Started

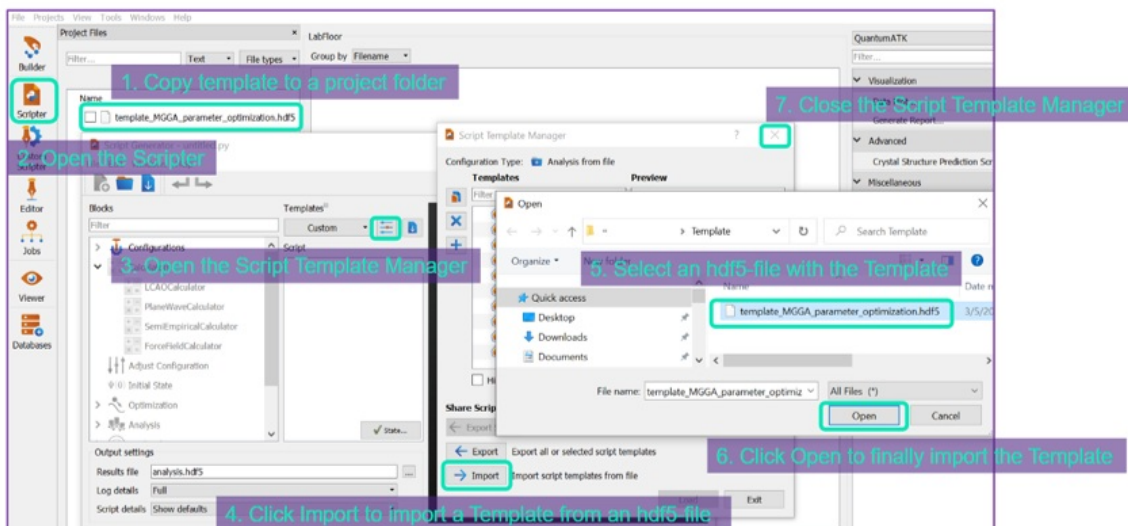
Create a new project folder in **NanoLab** and copy the following files containing different bulk phase configurations of crystalline  $\text{HfO}_2$  ([cubic.hdf5](#), [monoclinic.hdf5](#), [orthorhombic.hdf5](#)).

#### Tip

Crystal reference geometries can be conveniently obtained using the open databases such as Materials Project that is accessible via NanoLab

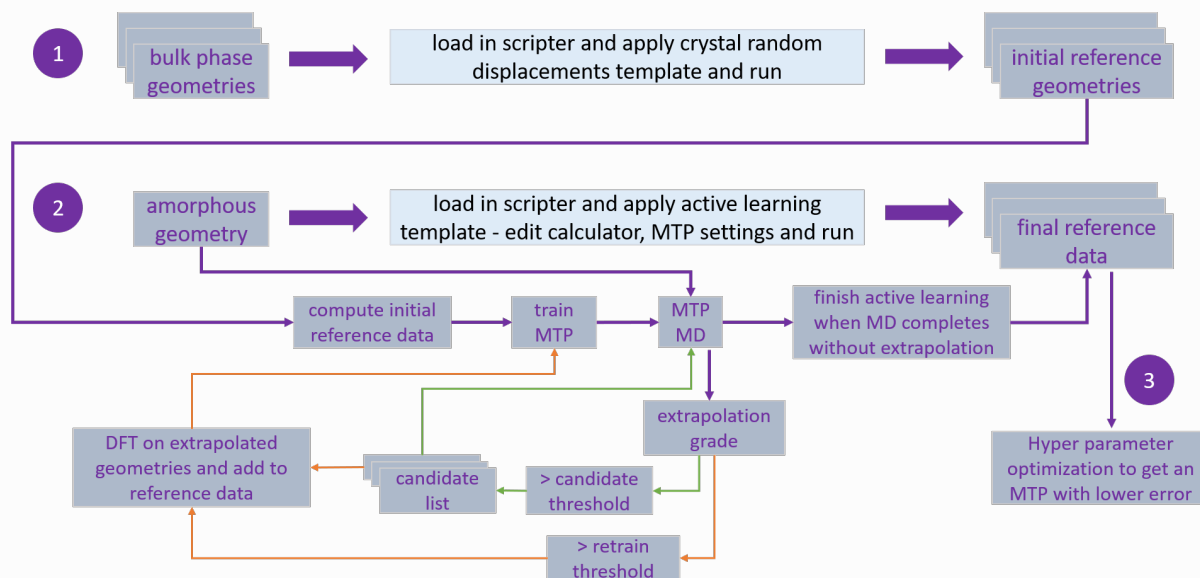
As the next step, import the following two templates [random\\_displacement\\_template.hdf5](#) and [active\\_learning\\_template.hdf5](#) to the script generator  following the procedure given in the below graphic.

# Importing a script template

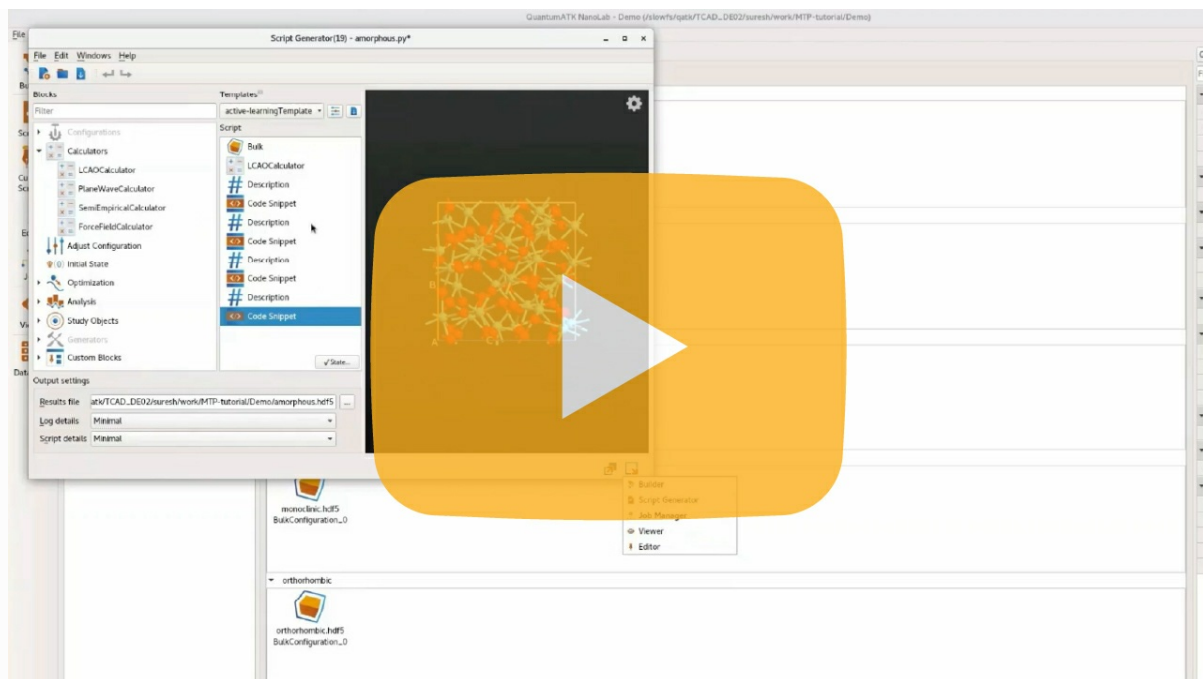


## Workflow


An schematic illustration of the workflow is shown below. There are three steps involved which are detailed in the text further below.



A video demonstrating the workflow can be viewed below.



## Step 1: Prepare Initial Reference Configurations

Load one of the bulk phase configurations of  $\text{HfO}_2$  on to the script generator  and apply the *random\_displacement\_template*. This will add a code snippet for `crystalTrainingRandomDisplacements` in which the user can edit the supercell repetition list, ionic and cell rattle amplitudes to create random configurations based on the geometry. These randomly generated geometries will be stored as a list in the file named *initial\_training\_sets.hdf5*. Choose large repetitions for small unitcells and *vice-versa*. Save the script and run it locally as it takes only a few seconds.

Repeat the above procedure for the other two bulk phase configurations of  $\text{HfO}_2$ . The corresponding python scripts used for this tutorial can be downloaded here ([cubic.py](#), [orthorhombic.py](#) and [monoclinic.py](#)). Notice that, in the above scripts, the supercell repetition list is different for the different crystal phases depending on the size of the unitcell. The template uses (1x1x1) and (2x2x1) supercells by default. However, for the cubic phase, include (2x2x2) and (3x3x3) instead of the default as in the scripts above. The resulting *initial\_training\_sets.hdf5* file will contain three lists one for each bulk phase.


### Note

The above scripts must be run in sequence and can be run locally (via `atkpython` in terminal or via JobManager) as it takes only a few seconds.

### Warning

If you alter the scripts from above, different repetition list or rattling amplitudes, then the training geometries will be different and ultimately result in a different MTP fitting parameter set. Thus, the users might find that the exact training and testing errors differ a bit from what is given further below.

## Step 2: Compute Reference Data and Setup Active Learning

Now that we have a set of initial reference geometries from step 1, we can proceed to compute reference data using DFT and improve the initial training data using active learning molecular dynamics on bulk amorphous configuration of  $\text{HfO}_2$ . For this, use the amorphous configuration from this file [amorphous.hdf5](#). This initial amorphous configuration was generated using the [Packmol Builder plugin](#) and preoptimized with `Trinastic_HfOSiTaTi_2013` forcefield. Send this configuration to the script generator  and apply active learning template. This loads a calculator and 4 code snippet blocks to the script generator that can be edited.

## Calculator: The reference calculator to use for MTP training

For this example, the DFT settings we have chosen include LCAO PseudoDojo medium basis set and PBE exchange-correlation functional. Detailed calculator settings can be found in the script [active-learning.py](#).

### Note

For accurate MTP training, higher accuracy settings than the ones used in this tutorial may be needed.

## Snippet 1: Import reference geometries

In the first snippet, there is code to extract the training data parameters for the reference configurations automatically from *initial\_training\_sets.hdf5*. The snippet also includes optional code to extract geometries from a [Trajectory](#) or [MDTrajectory](#) objects, which can be uncommented if one wants to use configurations from MD trajectories as initial training data.

## Snippet 2: Compute reference data

In the second snippet, we use [MomentTensorPotentialTraining](#) object to compute reference energies, forces and stress using the DFT reference calculator loaded earlier. The user can modify the values for *number\_of\_processes\_per\_task*, to set the number of MPI processes used for each reference calculation and for the *log\_filename\_prefix*. The other settings can be left as such. This snippet will create a file named *reference-data.hdf5* which includes the initial reference data set corresponding to the geometries from step 1 to be used in active learning.

### Note

We suggest running this script on a node with several cores as this could take 1 to 2 days using 16 cores.

## Snippet 3: Customize MTP training

In the third snippet, the user can set values of key parameters needed for the MTP training. This includes the following:

**NonLinearCoefficientsParameters** : User can specify if they like to perform optimization of the non linear coefficients that determine the shape of the basis functions for each element pair. If set to *True*, then there are options to perform optimization using energy only, maximum optimization iterations and a random seed to initialize the coefficients. Please note that this is a local optimization in the MTP parameter space, and different initial values may result in different solutions. How to find initial values for a good fit will be discussed below.

*constant\_terms*: No need to edit this part of the snippet as it automatically computes the isolated atomic energies of the elements in the reference data set using the reference calculator, which is not yet automatically computed for active learning.

**MomentTensorPotentialFittingParameters**: In this part of the snippet, the size of the basis set to be used, cutoff radii for atomic neighborhood description and force cap on configurations can be provided. Note that geometries with a maximum atomic force above the *forces\_cap* value will be ignored during training.

## Snippet 4: Set up active learning

In this final snippet all settings relevant for active learning will be provided.

**ActiveLearningSimulation**: In this part, user can provide the threshold values for the extrapolation grades and an interval at which they will be computed in MD [4]. The configuration will be added to a candidate list when the extrapolation grade goes above *candidate\_threshold* value and the MD will stop if the extrapolation grade goes over *retrain\_threshold*. The *max\_forces\_check* value sets the force value, which, when encountered during the simulation, will trigger computation of extrapolation grade for the

corresponding MD configuration.

**MaxwellBoltzmannDistribution**: is included to initialize the velocities at a target temperature for MD. In this tutorial example, we chose to run active learning at 3000 K.

**NPTMartynaTobiasKlein**: is used as the default MD ensemble method which relaxes both ions and volume. This can be changed to other ensemble methods if needed.

The method `runMolecularDynamics` starts an active learning MD simulation of the amorphous  $\text{HfO}_2$  geometry. Every time the `retrain_threshold` is reached, this method will be restarted until the MD simulation completes the set number of time steps. Before each restart, unique geometries from the candidate list will be selected and reference calculations on those geometries will be performed using the reference calculator. The additional reference data is then added to the initial dataset and a new MTP will be generated using the settings in snippet 3. This updated MTP will be used for the restarted MD simulation.

The script that is used for this tutorial is available here [📄 active-learning.py](#).

At the end of active learning simulation, a file named `active_learning_candidates.hdf5` is created, that contains the additional reference data.

#### Note

A common source of problems in MTP active learning is the following error message:

“No new candidates found in active learning MD.”

This is often accompanied by large values for the extrapolation grade. The reason for this problem is in most cases that the condition number of the training matrix is too large, which causes numerical inaccuracies when inverting this matrix to calculate the extrapolation grade and selecting the candidates via the maxvol algorithm. In this case it usually helps to reduce the number of MTP basis functions to improve the condition number and the numerical stability of the maxvol calculation. Alternatively, one can also try to include more different training configurations in the initial training set.

To repeat active learning simulation with a different MD setting, the training data computed so far should be put-together as the initial training data. In this case, `reference-data.hdf5` consists of the crystalline random displacement data as a **MomentTensorPotentialTraining** object and `active_learning_candidates.hdf5` contains data for unique amorphous geometries found in active learning as a **Trajectory** object. The `initial_training_data` parameter in the **ActiveLearningSimulation** block in the script should be a list of either **Trajectory** objects or **MomentTensorPotentialTraining** objects. We can convert a **MomentTensorPotentialTraining** object in file `reference-data.hdf5` into a **Trajectory** object and prepare the updated initial training data for consecutive active learning simulations as

```
moment_tensor_potential_training=nlread('reference-data.hdf5',MomentTensorPotentialTraining)[-1]
traj_1=TrainingSet(moment_tensor_potential_training, recalculate_training_data=False)
traj_1=traj_1.configurations()

traj_2=nlread('active_learning_candidates.hdf5',Trajectory)[-1]

initial_training_data = [traj_1, traj_2]
```

In a similar way, the `initial_training_data` can be augmented from several active learning simulations.

## Step 3: Find an MTP with Lowest Error

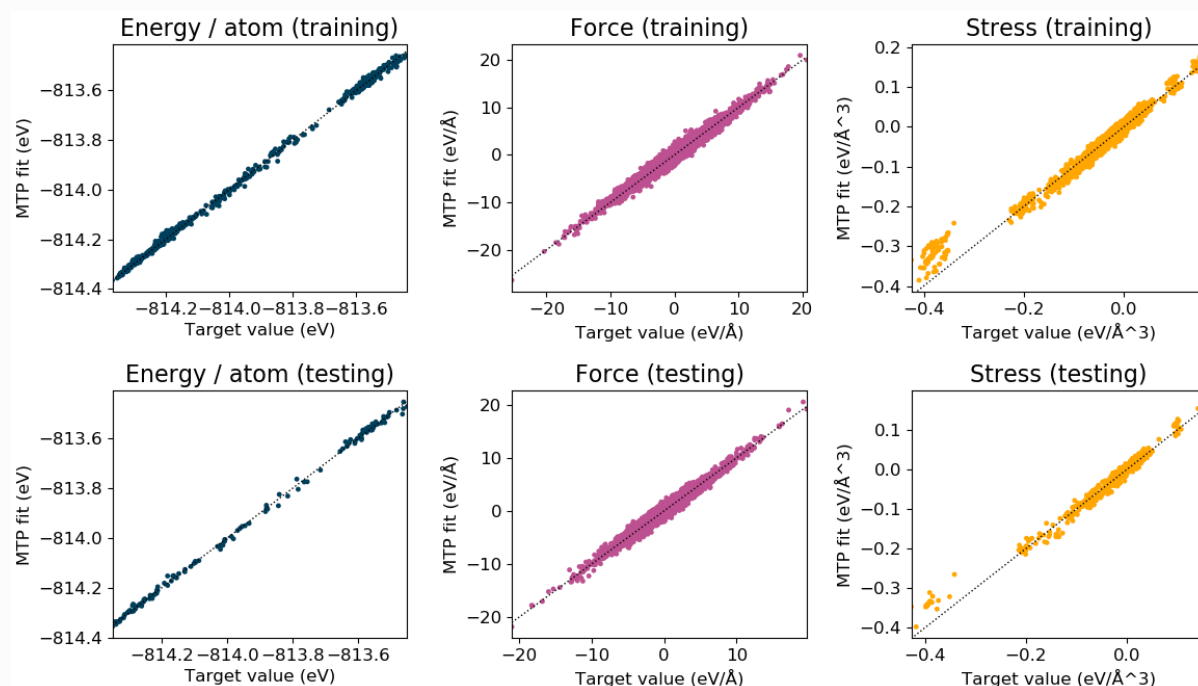
In active learning, the non linear coefficients are randomly initialized. So, in order to find an MTP with the lowest error, we have to train several MTPs with different initial non linear coefficients. This is synonymous with randomly sampling several points in the MTP hyper parameter space and performing a local optimization of the non linear coefficients.

This can be done using a script like this [MTP\\_training.py](#) which uses a for loop over different *random\_seed* value used to initialize the coefficients. Training data from the files *reference-data.hdf5* and *active\_learning\_candidates.hdf5* are used as input in this batch training. This script also uses 80% of the reference data for training and 20% for testing.

#### Note

This step involves training about 200 MTPs in sequence and might take several hours. For better performance choose MPI over openMP, as the MTP training implementation primarily supports MPI parallelization. Users could reduce the number of MTPs trained in line 32 of the script. By default, the script runs a non-linear coefficients optimization for each fit. To accelerate the training, users can turn the optimization off by specifying *perform\_optimization=False* and *initial\_coefficients=Random*. Then the fits run without local optimization of the non-linear coefficients and the user can choose to separately optimize the coefficients for the best fit from the 200.

The log output lists training errors for all of the MTPs generated from running the script and the MTP with lowest error for the training/test set (Energy RMSE: 10.7/10.7 meV/atom; Force RMSE: 0.39/0.40 eV/Å; stress RMSE:10.5/9.5 meV/Å<sup>3</sup>) is chosen for the validation MD test. Scatter plots comparing the energy, force and stress computed from MTP and reference method are shown below.



#### Note

Please note that training errors and the above plot will look different when any change to the scripts in steps 1, 2 and 3 are made. As long as the errors are comparable to the above given values, the MTP can be used in validation MD.

The MTP stated above can be generated using this script [MTP\\_training\\_best.py](#), provided the same reference data obtained from executing [active-learning.py](#) is used.

#### Note

In the above script, the random seed for initializing the non linear coefficients is chosen as that which gave the lowest force RMSE when running *MTP\_training.py* script. It is an optional step since the final mtp file is available from the output of *MTP\_training.py* anyways.

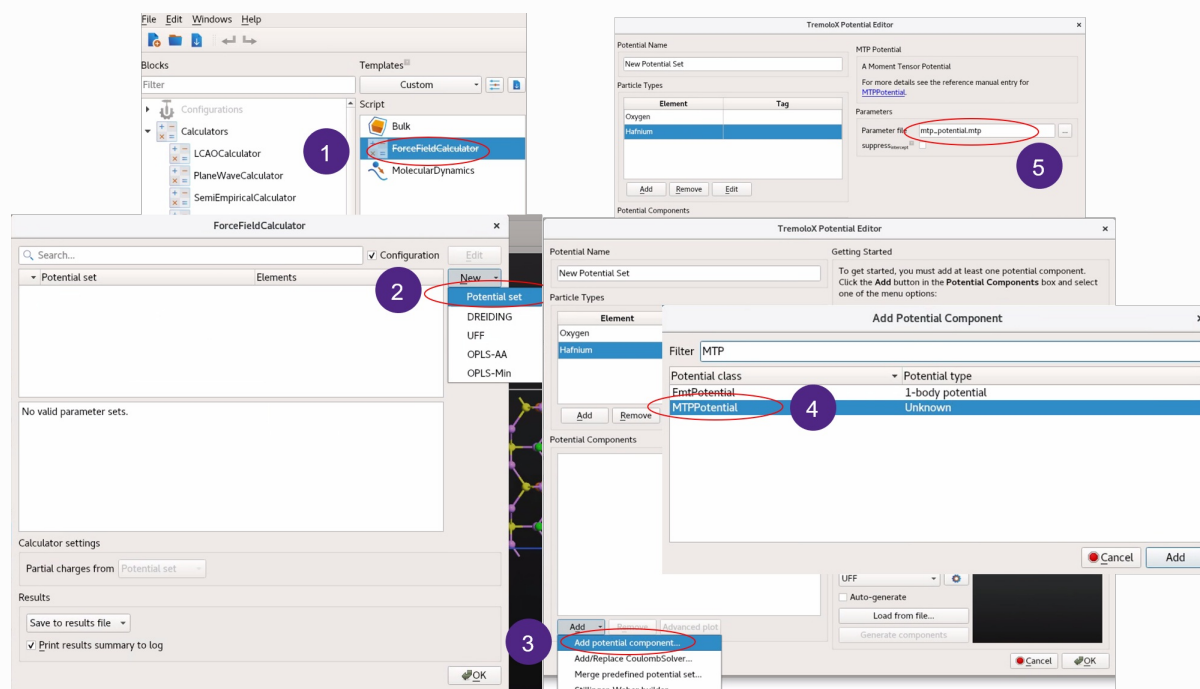


The scatter plot shown above can be created by running atkpython in a terminal and calling the below lines of code while inside the folder with the results for 'MTP\_training\_best.py'.

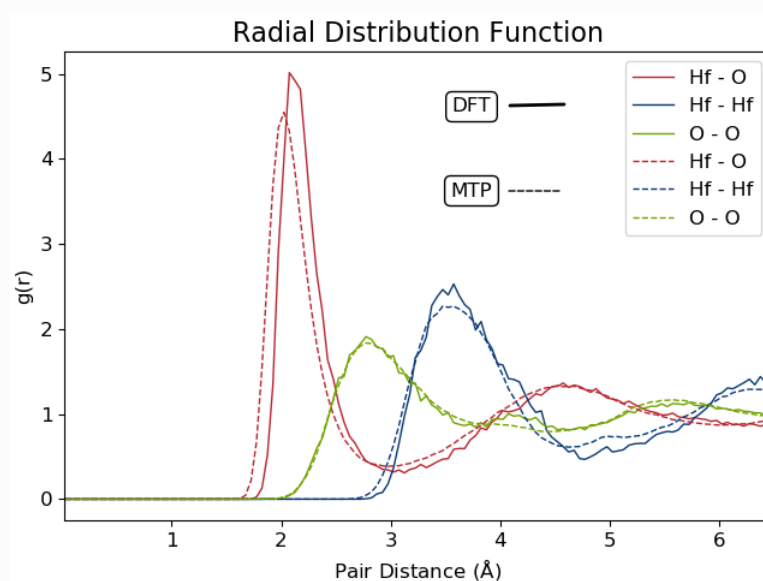
```
moment_tensor_potential_training=nlread('MTP_training.hdf5',MomentTensorPotentialTraining)[-1]
moment_tensor_potential_training._nlplotScatter(fit_index=0)
```

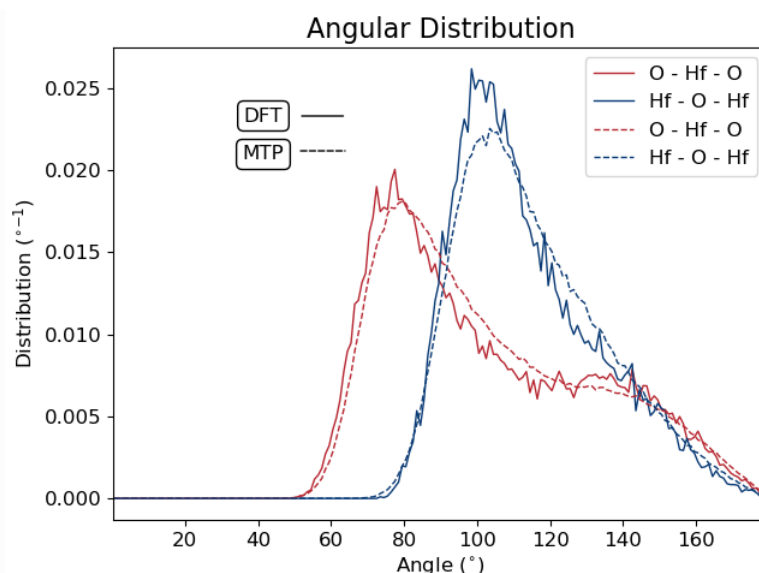
## Validation MD Simulation

Now, the MTP obtained in Step 3 will be employed in an MD simulation of amorphous HfO<sub>2</sub> for 0.1 ns using this script [MD.py](#). The mtp file used in this script can be found here [mtp\\_potential.mtp](#). This mtp parameter file can be loaded to the script generator via GUI as shown below (click the graphic to see enlarged).



The radial and angular distribution functions from the resulting trajectory is compared with those from a DFT-MD simulation using the same XC functional but reduced basis set as shown below.





As can be seen from the images above, there is a good agreement between the MTP generated trajectory and the DFT trajectory.

The above procedure can be directly applied to train accurate MTPs describing any bulk system of interest with minimal effort using QuantumATK.

## References

[1]

Yunxing Zuo, Chi Chen, Xiangguo Li, Zhi Deng, Yiming Chen, Jörg Behler, Gábor Csányi, Alexander V. Shapeev, Aidan P. Thompson, Mitchell A. Wood, and Shyue Ping Ong. Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, 124(4):731–745, 2020. URL: <https://doi.org/10.1021/acs.jpca.9b08723>, arXiv:<https://doi.org/10.1021/acs.jpca.9b08723>, doi:[10.1021/acs.jpca.9b08723](https://doi.org/10.1021/acs.jpca.9b08723).

[2]

Alexander V Shapeev. Moment tensor potentials: a class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016.

[3]

Evgeny V. Podryabinkin and Alexander V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Comput. Mater. Sci.*, 140:171–180, 2017. URL: <https://www.sciencedirect.com/science/article/pii/S0927025617304536>, doi:<https://doi.org/10.1016/j.commatsci.2017.08.031>.

[4] (1,2)

Konstantin Gubaev, Evgeny V. Podryabinkin, Gus L.W. Hart, and Alexander V. Shapeev. Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Comput. Mater. Sci.*, 156:148–156, 2019. URL: <https://www.sciencedirect.com/science/article/pii/S0927025618306372>, doi:<https://doi.org/10.1016/j.commatsci.2018.09.031>.

## FAQ Section

**Q1. In step 1, one prepares the initial configurations for their system. How important is it to define all of the different phases of the material since in the next step (during active learning), the missing configurations are added anyways?**

A1. Active learning is used to find and add missing configurations in the initial dataset with the objective to map out the configurational space of interest. But for active learning to work well, the initial training data should have at least coarsely sampled all relevant regions of the configurational space of interest to some extent. So, for example, if your interest is to get the properties of one particular crystalline phase of HfO<sub>2</sub> only – then it is enough to include the randomly displaced crystalline geometries of that crystalline phase only in step 1. Then the active learning, using the chosen crystalline phase as the initial geometry

for molecular dynamics, ensures finding missing geometries to represent that crystalline phase accurately. Now, if describing amorphous  $\text{HfO}_2$  is your goal, then it helps to include crystal displacement data for all available phases in step 1 and do the active learning as described in the tutorial.

## Q2. Is step 1 always necessary or can one start from active learning directly?

A2. If one looks at the workflow figure in the tutorial, it mentions training a preliminary MTP from the initial training data and use it to run the active learning MD (not DFT MD). DFT is employed only when missing geometries are found and needed to be added to the training data. So, we want to make sure that this preliminary MTP is already good enough to sample the equilibrium around the crystal geometries. This is where step 1 helps us immensely and becomes necessary.

## Q3. This question is about the fitting parameter options in step 2, snippet3. Is there some way to guess these options for particular system - for example cutoff radii (check if they have any sense) without actually finishing step 2.

A3. In snippet 3, there are three blocks. The constant term block is parameter free. Keywords in non-linear optimization block are also mostly universal. System specific keywords are in the MTP training block. Outer radii can be chosen based on the radial distribution function of the system and how many coordination shells you would like to include in the atomic description. Basis set size should be chosen based on the complexity of the atomic environment. Our suggestion is to start active learning with a smaller basis set (around 300 functions) and repeat it with increased basis. Step 2 includes computing the initial training geometry and active learning in which the former is the computationally expensive part. Once the initial training data is computed, re-running the script in the same folder will use the already computed initial reference data and only perform active learning part again. The costly step in active learning is the DFT calculation of the missing geometries. So every time you repeat active learning, the active learning candidates from previous run can be added to the initial training data set.

## Q4. Under non-linear parameters optimization, what else could be optimized if energy\_only option is False and when this is recommended?

A4. When this keyword is set to False, then the non-linear parameters are optimized with respect to E, F and stresses. We suggest using energy only optimization in active learning. If needed, you can optimize on forces and stresses in batch learning. Note that this is only a local optimization in the hyper parameter space.

## Q5. In the user example for ActiveLearningSimulations in the manual, Hf and O isolated energies are given in the example script but here, in the tutorial, we don't have them defined. Why is that?

A5. In snippet 3, there is a code block to compute the constant terms, which are the atomic energies of the elements in the system. They are computed automatically before training the preliminary MTP for active learning.

## Q6. In step 3, it is said that this may take several hours to get done. Where could one define the parallel options? Would it be possible to do in parallel the steps from the loop?

A6. Yes, one can run the script in step 3 multiple times with different random state while keeping the index in the loop smaller. Then consolidate the results from all the individual log files to find the one with the lowest error.

## Q7. Regarding the MTP\_training.py and MTP\_training\_best.py, the former contains a loop over random seeds values, while in the second one the value is given. Is the best (lowest RMSE) seed chosen automatically (given somewhere in main output of MTP\_training.py) or user need to actually screen all these mtp\_fit\_mtp\_i.log files?

A7. In this example, the best random seed is not found automatically. The output of MTP\_training.py (MTP\_training.log) contains a list of RMSEs and the corresponding MTP file names at the end of the file.

One can choose the .mtp file with the lowest force error and use it for validation MD. So, it is optional to run MTP\_training\_best.py. However, one can write a print statement (print(i,rand)) in line 34 of the script MTP\_training.py to get the values of 'rand' printed to the log file.

**Q8. This question is about validation MD simulation step. MD.py script contains MTP obtained in the step 3. But on the last two pictures there displayed also results of DFT-MD calculations. So what basis sets were used for the DFT-MD run?**

A8. For validation, a short DFT-MD with PBE XC functional and FHI Single Zeta Polarized basis set is performed to get the DFT results.

[← Previous](#)

[Next →](#)

---

© Copyright 2023 Synopsys, Inc. All Rights Reserved.