1 Introduction

NVNMD stands for non-von Neumann molecular dynamics.

Any user can follow two consecutive steps to run molecular dynamics (MD) on the proposed NVNMD computer, which has been released online: (i) to train a machine learning (ML) model that can decently reproduce the potential energy surface (PES); and (ii) to deploy the trained ML model on the proposed NVNMD computer, then run MD there to obtain the atomistic trajectories.

2 Preparation

2.1 Downloading source code

First, please visit https://github.com/LiuGroupHNU/nvnmd to download training and testing code for NVNMD.

Or get the source code with git:

```
$   cd /some/workspace
$   git clone https://github.com/LiuGroupHNU/nvnmd.git mynvnmd
$   cd mynvnmd
```

where **"mynvnmd"** is the name of the directory you wish to create on your machine.

Now we assume that **"$nvnmd_source_dir"** is the path to the current directory.

2.2 Installing the python interface

Check the python version on your machine by:

```
$   python --version
```

You can follow the virtual environment approach to install the tensorflow's Python interface. The full instruction can be found on the tensorflow's official website (available at https://www.tensorflow.org/install/pip). Now we assume that the Python interface will be installed to virtual environment directory "$tensorflow_venv":

```
$   virtualenv -p python3 $tensorflow_venv
$   source $tensorflow_venv/bin/activate
$   pip install --upgrade pip
$   pip install tensorflow==2.3.0
```

It is notice that everytime a new shell is started and one wants to use NVNMD, the virtual environment should be activated by:

```
$   source $tensorflow_venv/bin/activate
```

If one wants to skip out of the virtual environment, he/she can do:

```
$   deactivate
```

If one has multiple python interpreters named like python3.x, it can be specified by, for example:

```
$   virtualenv -p python3.7 $tensorflow_venv
```

If one does not need the GPU support and is concerned about package size, the CPU-only version of tensorflow should be installed by:

```
$   pip install tensorflow-cpu==2.3.0
```

One should remember to activate the virtual environment every time he/she uses NVNMD.

Then, execute

```
$   cd $nvnmd_source_dir
$   pip install .
```

You can print the help information by:

```
$   dp -h
```

2.3 Installing the C++ interface

Check the compiler version on your machine by:

```
$   gcc --version
```

The C++ interface was tested with compiler gcc >= 4.8.

First the C++ interface of Tensorflow should be installed. It is noted that the version of Tensorflow should be in consistent with the python interface.

The tensorflow's C++ interface will be compiled from the source code. Firstly, one installs bazel. The bazel version 3.1.0 should be used.

```
$   cd /some/workspace
$   wget
    https://github.com/bazelbuild/bazel/releases/download/3.1.0/bazel-
    3.1.0-installer-linux-x86_64.sh
$   chmod +x bazel-3.1.0-installer-linux-x86_64.sh
$   ./bazel-3.1.0-installer-linux-x86_64.sh --prefix /some/workspace/bazel
$   export PATH=/some/workspace/bazel/bin:$PATH
```

Then get the source code of the tensorflow:

```
$   git clone https://github.com/tensorflow/tensorflow tensorflow \
    -b v2.3.0 --depth=1
$   cd tensorflow
$   ./configure
```

Build the shared library of tensorflow:

```
$   bazel build -c opt --verbose_failures //tensorflow:libtensorflow_cc.so
```

You may want to add options `--copt=-msse4.2`, `--copt=-mavx`, `--copt=-mavx2` and `--copt=-mfma` to enable SSE4.2, AVX, AVX2 and FMA SIMD accelerations, respectively. It is noted that these options should be chosen according to the CPU architecture. If the RAM becomes an issue of your machine, you may limit the RAM usage by using --local_resources 2048,.5,1.0.

We assume that you want to install tensorflow in directory $tensorflow_root. Create the directory if it does not exists:

```
$   mkdir -p $tensorflow_root
```

Copy the libraries to the tensorflow's installation directory:

```
$   mkdir -p $tensorflow_root/lib
$   cp -d bazel-bin/tensorflow/libtensorflow_cc.so* $tensorflow_root/lib/
$   cp -d bazel-bin/tensorflow/libtensorflow_framework.so*
    $tensorflow_root/lib/
$   cp -d $tensorflow_root/lib/libtensorflow_framework.so.2
    $tensorflow_root/lib/libtensorflow_framework.so
```

Then copy the headers:

```
$   mkdir -p $tensorflow_root/include/tensorflow
$   rsync -avzh --exclude '_virtual_includes/' --include '*/' --include '*.h' -
    -include '*.inc' --exclude '*' bazel-bin/ $tensorflow_root/include/
$   rsync -avzh --include '*/' --include '*.h' --include '*.inc' --exclude '*'
    tensorflow/cc $tensorflow_root/include/tensorflow/
$   rsync -avzh --include '*/' --include '*.h' --include '*.inc' --exclude '*'
    tensorflow/core $tensorflow_root/include/tensorflow/
$   rsync -avzh --include '*/' --include '*' --exclude '*.cc' third_party/
    $tensorflow_root/include/third_party/
$   rsync -avzh --include '*/' --include '*' --exclude '*.txt' bazel-
    tensorflow/external/eigen_archive/Eigen/
    $tensorflow_root/include/Eigen/
$   rsync -avzh --include '*/' --include '*' --exclude '*.txt' bazel-
    tensorflow/external/eigen_archive/unsupported/
    $tensorflow_root/include/unsupported/
$   rsync -avzh --include '*/' --include '*.h' --include '*.inc' --exclude '*'
    bazel-tensorflow/external/com_google_protobuf/src/google/
    $tensorflow_root/include/google/
$   rsync -avzh --include '*/' --include '*.h' --include '*.inc' --exclude '*'
    bazel-tensorflow/external/com_google_absl/absl/
    $tensorflow_root/include/absl/
```

Now goto the source code directory of NVNMD and make a build place.

```
$   cd $nvnmd_source_dir/source
$   mkdir build
$   cd build
```

We assume that “$nvnmd_root” is the path to install NVNMD, then execute cmake

```
$   cmake -DTENSORFLOW_ROOT=$tensorflow_root \
    -DCMAKE_INSTALL_PREFIX=$nvnmd_root ..
```

If the cmake has executed successfully, then

```
$   make -j4
$   make install
```

The option “-j4” means using 4 processes in parallel. A different number can be used according to your hardware.

If everything works fine, you can use not only NVNMD but also DeePMD-kit.

3 Training

Our training procedure consists of not only the CNN training, but also the QNN training which uses the results of CNN as inputs. It is performed on CPU or GPU by using the training codes we open-sourced online.

To train a ML model that can decently reproduce the PES, training and testing data set should be prepared first. This can be done by using either the state-of-the-art active learning tools, or the outdated (i.e., less efficient) brute-force density functional theory (DFT)-based ab-initio molecular dynamics (AIMD) sampling.

Then, copy the data set to working directory:

```
$   cp -r $dataset $nvnmd_workspace/data
```

where **“$dataset”** is the path to the data set and **“$nvnmd_workspace”** is the path to working directory. “$nvnmd_source_dir/examples/nvnmd/data” is the path to the data set used in this example.

Remember to activate the virtual environment by:

```
$   source $tensorflow_venv/bin/activate
```

3.1 CNN training

3.1.1 Input script

Goto the working directory and make a training place:

```
$   cd $nvnmd_workspace
$   mkdir ws-1
$   cd ws-1
```

where **“ws-1”** is the name of the directory for training.

Then create a directory for CNN training and copy the input script to new directory:

```
$   mkdir s1
$   cd s1
$   cp $nvnmd_source_dir/examples/nvnmd/train-1.json train.json
```

where **“s1”** is the name of the directory for CNN training and **“train-1.json”** is the name of input script for CNN training.

The structure of the input script is as follows:

```
{
	"model": {…},
	"nvnmd": {…},
	"learning_rate": {…},
	"loss": {…},
	"training": {…}
}
```

A model has two parts, a descriptor that maps atomic configuration to a set of symmetry invariant features, and a fitting net that takes descriptor as input and predicts the atomic contribution to the target physical property. It's defined in the model section, for example:

```
    "model": {
        "descriptor": {
            "seed": 1,
            "type": "se_a",
            "sel": [60, 60],
            "rcut": 7.0,
            "rcut_smth": 0.5,
            "neuron": [5, 10, 20],
            "type_one_side": true,
            "axis_neuron": 10,
            "resnet_dt": false
        },
        "fitting_net": {
            "seed": 1,
            "neuron": [20, 20, 20],
            "resnet_dt": false
        }
    },
```

- **“model/descriptor/type”** should be set to **“se_a”**.
- **“model/descriptor/sel”** gives the maximum possible number of neighbors in the cut-off radius. It is a list, the length of which is the same as the number of atom types in the system, and sel[i] denote the maximum possible number of neighbors with type i.
- **“model/descriptor/rcut”** is the cut-off radius for neighbor searching.
- **“model/descriptor/rcut_smth”** gives where the smoothing starts.
- **“model/descriptor/neuron”** should be set to **[5, 10, 20]**, which specifies the size of the embedding net. From left to right the members denote the sizes of each hidden layer from input end to the output end, respectively.
- **“model/descriptor/type_one_side”** should be set to **true** so that descriptor will only consider the types of neighbor atoms. Otherwise, both the types of centric and neighbor atoms are considered.

- **“model/descriptor/axis_neuron”** should be set to **10**, which specifies the size of submatrix of the embedding matrix, the axis matrix as explained in the DeepPot-SE paper (available at https://arxiv.org/abs/1805.09003).
- **“model/fitting_net/neuron”** should be set to **[20, 20, 20]**, which specifies the size of the fitting net.
- **“resnet_dt”** should be set to **false**, then a timestep is not used in the ResNet.

The nvnmd section is defined as follows:

```
"nvnmd":{
        "config_file":"none",
        "weight_file":"none",
        "map_file":"none",
        "enable":true,
        "restore_descriptor":false,
        "restore_fitting_net": false,
        "quantize_descriptor":false,
        "quantize_fitting_net":false
  },
```

- **“nvnmd/config_file”** is used to load the configuration file, which should be set to **“none”** for CNN training.
- **“nvnmd/weight_file”** is used to load the weight file, which should be set to **“none”** for CNN training.
- **“nvnmd/map_file”** is used to load the mapping table, which should be set to **“none”** for CNN training.
- **“nvnmd/enable”** is used to determine whether to use NVNMD, which should be set to **true**.
- **“nvnmd/restore_descriptor”** and **“nvnmd/restore_fitting_net”** is used to determine whether to restore the trained model and parameters, which should be set to **false** for CNN training.
- **“nvnmd/quantize_descriptor”** and **“nvnmd/quantize_fitting_net”** is used to determine whether to quantize the weights and activations, which should be set to **false** for CNN training.

You can modify the value of parameters in the input script as needed.

3.1.2 Training

- IN: train.json
- OUT: model.ckpt

CNN training can be invoked by:

```
$   dp train train.json
```

3.1.3 Freezing the model

- IN: model.ckpt
- OUT: graph.pb, nvnmd/weight.npy, nvnmd/config.npy

To freeze the model, typically one does:

```
$   dp freeze -o graph.pb -w nvnmd/weight.npy
```

where **"graph.pb"** is the name of frozen model file, **"weight.npy"** is the name of weight file.

3.1.4 Testing

The frozen model can be used in many ways. The most straightforward testing can be invoked by:

```
$   mkdir test-test
$   dp test -m ./graph.pb \
    -s /path/to/system -d ./test-test/detail \
    -n 999999999 | tee test-test/output
```

where the frozen model file to import is given via the **"-m"** command line flag, the path to the testing data set is given via the **"-s"** command line flag, the file containing details of energy, force and virial accuracy is given via the **"-d"** command line flag, the amount of data for testing is given via the **"-n"** command line flag.

3.1.5 Building the mapping table

- IN: nvnmd/config.npy, nvnmd/weight.npy
- OUT: nvnmd/map.npy

You can build the mapping table and then switch to the directory for training by:

```
$   dp map -c nvnmd/config.npy -w nvnmd/weight.npy \
    -m nvnmd/map.npy
$   cd ../
```

where **"map.npy"** is the name of mapping table.

3.2 QNN training

3.2.1 Input script

Create a directory for QNN training and copy the input script to new directory:

```
$   mkdir s2
$   cd s2
$   cp $nvnmd_source_dir/examples/nvnmd/train-2.json train.json
```

where **"s2"** is the name of the directory for QNN training and **"train-2.json"** is the name of input script for QNN training.

Compared with the input script for CNN training, some parameters for QNN training need to be modified. An example is provided as follows:

```
"nvnmd":{
        "config_file":"../s1/nvnmd/config.npy",
        "weight_file":"../s1/nvnmd/weight.npy",
        "map_file":"../s1/nvnmd/map.npy",
        "enable":true,
        "restore_descriptor":true,
        "restore_fitting_net":true,
        "quantize_descriptor":true,
        "quantize_fitting_net":true
 },

"learning_rate": {
    "start_lr": 0.0000005,
    …
},

"training": {
    "stop_batch": 10000,
    …
}
```

- **“nvnmd/config_file”** should be set to **“../s1/nvnmd/config.npy”**.
- **“nvnmd/weight_file”** should be set to **“../s1/nvnmd/weight.npy”**.
- **“nvnmd/map_file”** should be set to **“../s1/nvnmd/map.npy”**.
- **“nvnmd/restore_descriptor”** and **“nvnmd/restore_fitting_net”** should be set to **true** for QNN training to restore parameters obtained by CNN training.
- **“nvnmd/quantize_descriptor”** and **“nvnmd/quantize_fitting_net”** should be set to **true** for QNN training to quantize the weights and activations.

Typically, CNN training uses a large number of training steps with a high learning rate; and the subsequent QNN training uses a small number of training steps (e.g., 1×10^4) and a low learning rate (e.g., 5×10^{-7}), as it only needs to minimize the small error induced by quantization from CNN to QNN.

You can modify the value of parameters in the input script as needed.

3.2.2 Training

- IN: train.json
- OUT: model.ckpt

QNN training can be invoked by:

```
$   dp train train.json
```

3.2.3 Freezing the model

- IN: model.ckpt
- OUT: graph.pb, nvnmd/weight.npy, nvnmd/config.npy

To freeze the model, typically one does:

```
$   dp freeze -o graph.pb -w nvnmd/weight.npy
```

where **"graph.pb"** is the name of frozen model file, **"weight.npy"** is the name of weight file.

3.2.4 Testing

The frozen model can be used in many ways. The most straightforward testing can be invoked by:

```
$   mkdir test-test
$   dp test -m ./graph.pb \
    -s /path/to/system -d ./test-test/detail \
    -n 999999999 | tee test-test/output
```

where the frozen model file to import is given via the **"-m"** command line flag, the path to the testing data set is given via the **"-s"** command line flag, the file containing details of energy, force and virial accuracy is given via the **"-d"** command line flag, the number of data for testing is given via the **"-n"** command line flag.

3.2.5 Wrapping the ML model

- IN: nvnmd/config.npy, nvnmd/weight.npy, ../s1/nvnmd/map.npy
- OUT: nvnmd/model.pb

Then you can wrap the ML model by:

```
➢   dp wrap -c nvnmd/config.npy -w nvnmd/weight.npy \
    -m ../s1/nvnmd/map.npy -o nvnmd/model.pb
```

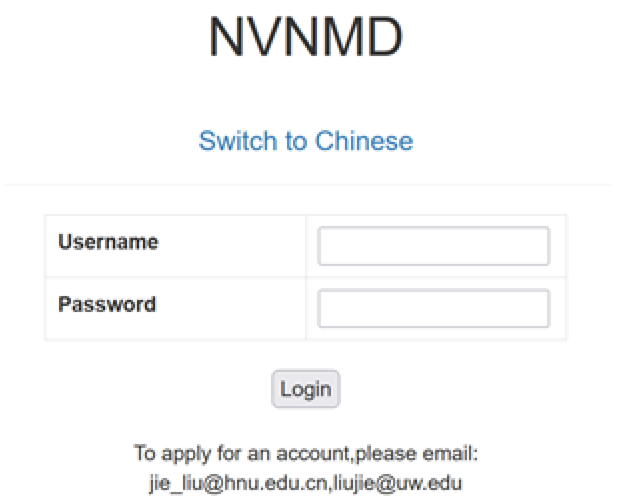
where **"model.pb"** is the name of ML model.

4 Running MD

After CNN and QNN training, you can upload the ML model to our online NVNMD system and run MD there.

4.1 Account application

The server website of NVNMD is available at http://nvnmd.picp.vip. You can visit the URL and enter the login interface (Figure.1).


NVNMD

Switch to Chinese

Username	
Password	

Login

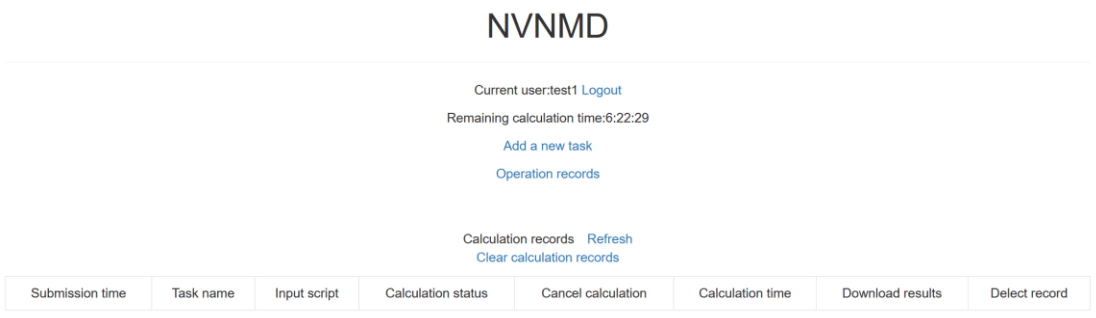
To apply for an account,please email:
jie_liu@hnu.edu.cn,liujie@uw.edu

(Figure.1 The login interface)

To obtain an account, please send your application to the email (jie_liu@hnu.edu.cn, liujie@uw.edu). The username and password will be sent to your by email.

4.2 Adding a new task

After successfully obtaining the account, enter the username and password in the login interface, and click “Login” to enter the homepage (Figure.2).


NVNMD

Current user:test1 Logout

Remaining calculation time:6:22:29

Add a new task

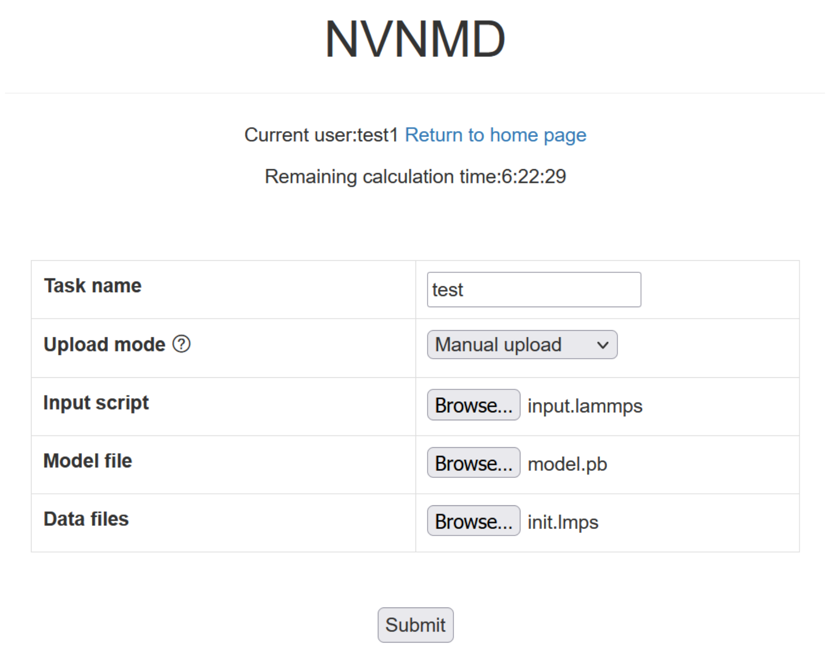
Operation records

Calculation records Refresh
Clear calculation records

Submission time	Task name	Input script	Calculation status	Cancel calculation	Calculation time	Download results	Delect record

(Figure.2 The homepage)

The homepage displays the remaining calculation time and all calculation records not deleted. Click "Add a new task" to enter the interface for adding a new task (Figure.3).


NVNMD

Current user:test1 Return to home page

Remaining calculation time:6:22:29

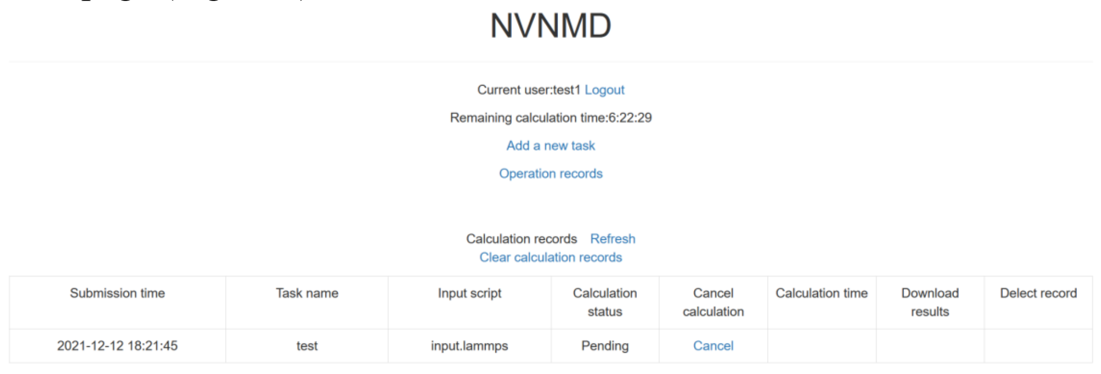
Task name	test
Upload mode ⓘ	Manual upload
Input script	Browse... input.lammps
Model file	Browse... model.pb
Data files	Browse... init.lmps

Submit

(Figure.3 The interface for adding a new task)

- **"Task name"**: name of the task.
- **"Upload mode"**: two modes of uploading results to online data storage, including "Manual upload" and "Automatic upload". Results need to be uploaded manually to online data storage with "Manual upload" mode, and will be uploaded automatically with "Automatic upload" mode.
- **"Input script"**: input file of the MD simulation.
  In the input script, one needs to specify the pair style as follows:
  ```
  pair_style fpga 7.0
  pair_coeff
  ```
  where **"7.0"** is global cutoff.
- **"Model file"**: the ML model obtained by QNN training.
- **"Data files"**: data files containing information required for running an MD simulation.

Click "Submit" to submit the task and then automatically return to the homepage (Figure.4).


NVNMD

Current user:test1 Logout

Remaining calculation time:6:22:29

Add a new task

Operation records

Calculation records Refresh

Clear calculation records

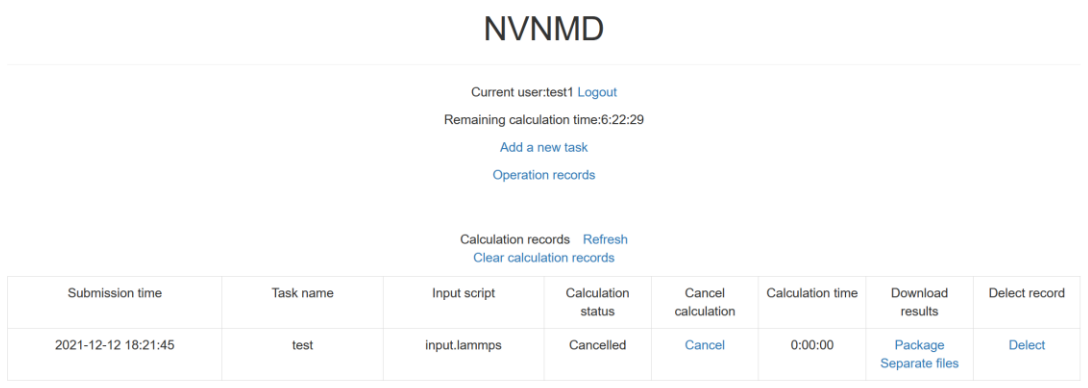
Submission time	Task name	Input script	Calculation status	Cancel calculation	Calculation time	Download results	Delect record
2021-12-12 18:21:45	test	input.lammps	Pending	Cancel			

(Figure.4 The homepage with a new record)

Click "Refresh" to view the latest status of all calculation tasks.

4.3 Cancelling calculation

For the task whose calculation status is "Pending" and "Running", you can click the corresponding "Cancel" on the homepage to stop the calculation (Figure.5).



NVNMD

Current user:test1 Logout

Remaining calculation time:6:22:29

Add a new task

Operation records

Calculation records Refresh

Clear calculation records

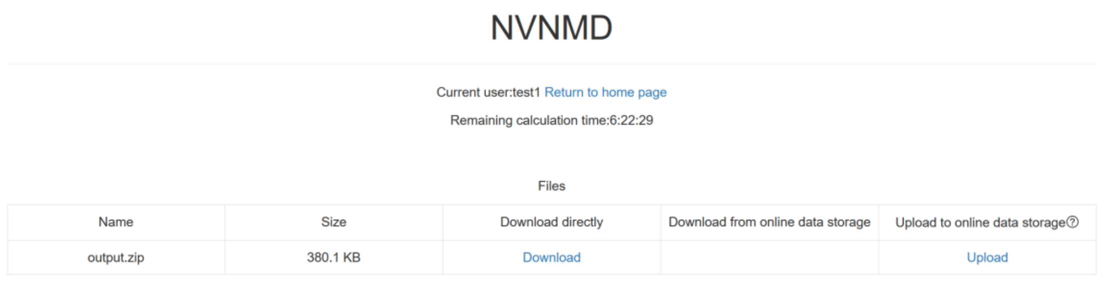
Submission time	Task name	Input script	Calculation status	Cancel calculation	Calculation time	Download results	Delect record
2021-12-12 18:21:45	test	input.lammps	Cancelled	Cancel	0:00:00	Package Separate files	Delect

(Figure.5 The homepage with a cancelled task)

4.4 Downloading results

For the task whose calculation status is "Completed", "Failed" and "Cancelled", you can click the corresponding "Package" or "Separate files" in the "Download results" bar on the homepage to download results.

Click "Package" to download a zipped package of all files including input files and output results (Figure.6).



NVNMD

Current user:test1 Return to home page

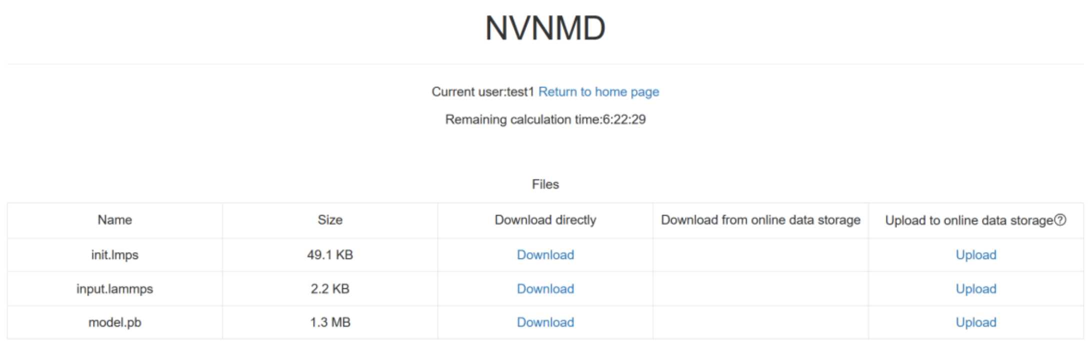
Remaining calculation time:6:22:29

Files

Name	Size	Download directly	Download from online data storage	Upload to online data storage⑦
output.zip	380.1 KB	Download		Upload

(Figure.6 The interface for downloading a zipped package)

Click "Separate files" to download the required separate files (Figure.7).



NVNMD

Current user:test1 Return to home page

Remaining calculation time:6:22:29

Files

Name	Size	Download directly	Download from online data storage	Upload to online data storage⑦
init.lmps	49.1 KB	Download		Upload
input.lammps	2.2 KB	Download		Upload
model.pb	1.3 MB	Download		Upload

(Figure.7 The interface for downloading separate files)

If "Manual upload" mode is selected or the file has expired, click

“Upload” on the download interface to upload manually.

4.5 Delete record

For the task no longer needed, you can click the corresponding “Delete” on the homepage to delete the record.

Records cannot be retrieved after deletion.

4.6 Clear records

Click “Clear calculation records” on the homepage to clear all records.

Records cannot be retrieved after clearing.