A Notebook-based Platform for Computational Chemistry and Materials Science

Chun-Yaung Lu
alu@tacc.texas.edu
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain **live code, equations, visualizations** and **narrative text**. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

https://jupyter.org/
• Use your TACC or XSEDE User Portal username and password to log in
• Run VNC, iPython/Jupyter Notebook, R Studio on Stampede2 or Wrangler
• Available Stampede2 queues for iPython/Jupyter: development, skx-dev, normal, skx-normal, ...
• One compute node each time
  KNL: 68 cores, SKX: 48 cores

https://vis.tacc.utexas.edu
Running LAMMPS

On Stampede2:

```bash
$module load lammps
```

will load default version 16Mar18.
The latest version 5Jun19 is also available

In Python3

```python
>>> from lammps import IPyLammps
L = IPyLammps()
```

LAMMPS output is captured by PyLammps wrapper

In Jupyter

```python
In [1]: from lammps import IPyLammps
L = IPyLammps()
LAMMPS output is captured by PyLammps wrapper
```

```python
In [2]: from lammps import PyLammps
L = PyLammps()
LAMMPS output is captured by PyLammps wrapper
```

[prints output]

https://lammps.sandia.gov/doc/Python_head.html
## More Packages

There are many many other packages that you can install and use in Jupyter

Many packages can be installed simply by using pip:

```
$ pip install myPackage --user
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpaw</td>
<td>1.5.1</td>
<td>Quantum DFT</td>
</tr>
<tr>
<td>lammps</td>
<td>22Aug18</td>
<td>Classical MD</td>
</tr>
<tr>
<td>hoomd-blue</td>
<td>2.3.5</td>
<td>Classical MD, CGMD</td>
</tr>
<tr>
<td>ase</td>
<td>3.17.0</td>
<td>Simulation interface</td>
</tr>
<tr>
<td>tsase</td>
<td>master</td>
<td>Transition state library for ASE</td>
</tr>
<tr>
<td>rdkit</td>
<td>2018_03_4</td>
<td>Cheminformatics, ML</td>
</tr>
<tr>
<td>mdtraj</td>
<td>1.9.2</td>
<td>Analysis tool</td>
</tr>
<tr>
<td>pytraj</td>
<td>2.0.3</td>
<td>Analysis tool</td>
</tr>
<tr>
<td>cpptraj</td>
<td>18.00</td>
<td>Analysis tool</td>
</tr>
<tr>
<td>parmed</td>
<td>3.03</td>
<td>Analysis tool</td>
</tr>
<tr>
<td>OpenKIM</td>
<td>1.9.7</td>
<td>Force field database</td>
</tr>
<tr>
<td>libxc</td>
<td>4.2.3</td>
<td>XC library</td>
</tr>
<tr>
<td>libvdwxc</td>
<td>0.3.2</td>
<td>XC-VDW library</td>
</tr>
<tr>
<td>nglview</td>
<td>1.1.7</td>
<td>Visualizer</td>
</tr>
</tbody>
</table>
# Atomic Simulation Environment (ASE) Supported Software

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>ST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asap</td>
<td>Highly efficient EMT code</td>
<td></td>
</tr>
<tr>
<td>GPAW</td>
<td>Real-space/plane-wave/LCAO PAW code</td>
<td>YES</td>
</tr>
<tr>
<td>Hotbit</td>
<td>DFT based tight binding</td>
<td></td>
</tr>
<tr>
<td>abinit</td>
<td>Plane-wave pseudopotential code</td>
<td></td>
</tr>
<tr>
<td>amber</td>
<td>Classical molecular dynamics code</td>
<td>YES</td>
</tr>
<tr>
<td>castep</td>
<td>Plane-wave pseudopotential code</td>
<td></td>
</tr>
<tr>
<td>cp2k</td>
<td>DFT and classical potentials</td>
<td></td>
</tr>
<tr>
<td>demon</td>
<td>Gaussian based DFT code</td>
<td></td>
</tr>
<tr>
<td>dftb</td>
<td>DFT based tight binding</td>
<td></td>
</tr>
<tr>
<td>dmol</td>
<td>Atomic orbital DFT code</td>
<td></td>
</tr>
<tr>
<td>QE</td>
<td>Plane-wave pseudopotential code</td>
<td>YES</td>
</tr>
<tr>
<td>exciting</td>
<td>Full Potential LAPW code</td>
<td></td>
</tr>
<tr>
<td>aims</td>
<td>Numeric atomic orbital, full potential code</td>
<td></td>
</tr>
<tr>
<td>fleur</td>
<td>Full Potential LAPW code</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>ST2</th>
</tr>
</thead>
<tbody>
<tr>
<td>gaussian</td>
<td>Gaussian based electronic structure code</td>
<td>YES</td>
</tr>
<tr>
<td>gromacs</td>
<td>Classical molecular dynamics code</td>
<td>YES</td>
</tr>
<tr>
<td>gulp</td>
<td>Interatomic potential code</td>
<td>YES</td>
</tr>
<tr>
<td>jcapo</td>
<td>Plane-wave ultra-soft pseudopotential code</td>
<td></td>
</tr>
<tr>
<td>lammps</td>
<td>Classical molecular dynamics code</td>
<td>YES</td>
</tr>
<tr>
<td>mopac</td>
<td>Semiempirical quantum chemistry code</td>
<td></td>
</tr>
<tr>
<td>nwchem</td>
<td>Gaussian based electronic structure code</td>
<td>YES</td>
</tr>
<tr>
<td>octopus</td>
<td>Real-space pseudopotential code</td>
<td></td>
</tr>
<tr>
<td>onetep</td>
<td>Linear-scaling pseudopotential code</td>
<td></td>
</tr>
<tr>
<td>siesta</td>
<td>LCAO pseudopotential code</td>
<td>YES</td>
</tr>
<tr>
<td>turbomol</td>
<td>Fast atom orbital code</td>
<td></td>
</tr>
<tr>
<td>VASP</td>
<td>Plane-wave PAW code</td>
<td>YES</td>
</tr>
<tr>
<td>dftd3</td>
<td>DFT-D3 dispersion correction calculator</td>
<td></td>
</tr>
</tbody>
</table>

[https://wiki.fysik.dtu.dk/ase/ase/calculators/calculators.html#module-ase.calculators](https://wiki.fysik.dtu.dk/ase/ase/calculators/calculators.html#module-ase.calculators)
LAMMPS + Jupyter + HPC

TACC Vis Portal lets you run Jupyter Notebook on ONE compute node (KNL: 68 cores, SKX: 48 cores)

- LAMMPS’s USER-OMP package (provides optimized and multi-threaded version of many LAMMPS functions)
  ```python
  In [18]:
  L.clear()
  L.package("omp 4")
  
  Out[18]: ['set 4 OpenMP thread(s) per MPI task', 'using multi-threaded neighbor list subroutines']
  ```

- Run LAMMPS on multiple processors (use ipyparallel + mpi4py)

**MPI4Py:**

MPI for Python provides bindings of the Message Passing Interface (MPI) standard for the Python programming language, allowing any Python program to exploit multiple processors.


**Ipyparallel:**

Ipyparallel (formerly IPython parallel) enables all types of parallel applications to be developed, executed, debugged, and monitored interactively.

Parallel Scheme (domain decomposition)

**Step 1**
Start ipcluster

**Step 2**
Import ipyparallel

**Step 3**
Import mpi4py and start LAMMPS

```
In [1]: import os
   ...: import ipyparallel as ipp
   ...: rc = ipp.Client(profile='mpi')
   ...: view = rc[:]
   ...: print("Total number of MPI tasks =",len(view))

Total number of MPI tasks = 4
```

Cell magic for executing python commands on the ipython engines

```
In [3]: %%capture
   ...: px
   ...: from mpi4py import MPI
   ...: from lamlamps import IPyLammps
   ...: L = IPyLammps(cmdargs="-log",logfile)
```
Parallel Scheme (multi-replica)

(Example: lammps_neb_hop1.ipynb)

E.g. 13 replicas, 2 cpus/replica

Start with 26 iPython engines

Start LAMMPS with 13 replicas with 2 cores/replica
Making Workflow: PARSL

Parsl is a native Python library. It allows you to write functions that execute in parallel and tie them together with dependencies to create workflows.

“App” is a piece of code that can be asynchronously executed on an execution resource

Parsl provides support for pure Python apps (python_app) and also command-line apps executed via Bash (bash_app)

Parsl creates implicit workflows based on the passing of control or data between Apps.

```python
@python_app
def hello():
    return 'Hello World!'

@bash_app
def echo_hello(stdout='echo-hello.stdout', stderr='echo-hello.stderr'):  
    return 'echo "Hello World!!"'

@bash_app
def run_lammps(stdout='stdout', stderr='lmp.stderr'):  
    return 'lmp_stampede < input'
```

http://parsl-project.org/
Example 1 (Fe/Cr random alloy)

Pure python code

Parallel workflow
Example 2 (AI surface diffusion, Hessian matrix)

Create structures of two minimum states

M2

M1

Nudged-elastic band

Saddle point structure

Compute Hessian 3n elements/worker

Calculate eigen values

Compute rate prefactor

$H_{3n \times 3n} = \{ \partial V / \partial x_i \partial x_j \}$

Pure python code

$H_{3n \times 3n} = \{ \partial V / \partial x_i \partial x_j \}$
Example 3 (Alkane C-H Bond dissociation Energy)

Alkane isomers:
E.g. C₅H₁₂

- n-pentane
- neo-pentane
- i-pentane

Number of isomers:
- C₁₀H₂₂: 75
- C₁₂H₂₆: 355
- C₁₅H₃₂: 4,347
- C₁₈H₃₈: 60,523
- C₂₀H₄₂: 366,319
- C₂₁H₄₄: 910,726
- C₂₄H₅₀: > 14.5M

Python script

Isomer structure code
40000 (neo-pentane)

Python script

SMILES code
C(C)(C)(C)(C)

openbabel

3D structure (PDB format)
x, y, z ...

Python script

LAMMPS input files

MD & Minimization (LAMMPS)

Remove H & Minimization (LAMMPS)

Python script

E₁

E₂

E₂ - E₁
Example 3 (performance test)

Stampede2 SKX node (48 cores)

48 workers
Efficiency ~50%
96 mins → 4 mins

Run directly
$time python3 alkane.py
Example 4 (Fe/Cr random alloy, ipywidgets)

Set Cr concentration → Create structure → Compute Energy → plot

In [3]:
```
im=interact_manual(Reset)
im.widget.children[0].description = 'Reset'
interactive_plot = interactive(GenStruct, p=(0, 1, 0.1))
output = interactive_plot.children[-1]
output.layout.height = '350px'
interactive_plot
```

Reset

p  0.20

Cr/Fe random alloy
Cr = 20.0 %
Energy = -1188.093495 eV