Using Python in LAMMPS

Dr. Richard Berger

Temple University

LAMMPS Workshop 2017
Outline

Modes

Installation

Computes / Function Calls

Pair Python

Fix Python

Library Interface

Jupyter notebooks
Modes

Embedded Python
- LAMMPS binary launches embedded Python interpreter
- executes Python code on demand

Python as Driver
- Python starts LAMMPS as library
- controls LAMMPS through library calls
Modes

Hybrid (Embedded)
- Python code launched from LAMMPS can control/access it like a library

```
LAMMPS
  └── libpython
```

Hybrid (Python as Driver)
- LAMMPS can call Python code defined in driver and has access to all global objects

```
Python
  └── liblammmps
```
Requirements

- PYTHON package must be installed
- LAMMPS must be compiled as **shared-library**
- (Optional) `-DLAMMPS_EXCEPTIONS` for better error handling
- LAMMPS Python module (lammps.py) must be installed
Exception Traceback (most recent call last)
<ipython-input-3-6bfbffdb2363> in <module>()
----> 1 L.test()

~GitHub/lammps/lammps/build_py3/myenv3/lib/python3.6/site-packages/lammps.py in handler(*args, **kwargs)
    810     with OutputCapture() as capture:
    811         self.command(' '.join(cmd_args))
--> 812         output = capture.output
    813
    814

~GitHub/lammps/lammps/build_py3/myenv3/lib/python3.6/site-packages/lammps.py in command(self, cmd)
    625     def command(self, cmd):
    626         self.lmp.command(cmd)
--> 627         self._cmd_history.append(cmd)
    628
    629

~GitHub/lammps/lammps/build_py3/myenv3/lib/python3.6/site-packages/lammps.py in command(self, cmd)
    193         if error_type == 2:
    194             raise MPIAbortException(error_msg)
--> 195             raise Exception(error_msg)
    196
    197     # send a list of commands

Exception: ERROR: Unknown command: test (/home/richard/GitHub/lammps/lammps/src/input.cpp:315)
Step 1: Building LAMMPS as a shared library

```
cd $LAMMPS_DIR/src

make yes-PYTHON

# compile shared library
make mpi mode=shlib LMP_INC="-DLAMMPS_EXCEPTIONS"
```
Step 2: Installing the LAMMPS Python package

```
cd $LAMMPS_DIR/python
python install.py
```

**Warning!**

Recompiling the shared library requires reinstalling the Python package.
CMake (soon)

```bash
$ cd $LAMMPS_DIR/src
mkdir build
$ cd build
$ cmake ../cmake -DENABLE_PYTHON=on \  -DBUILD_SHARED_LIBS=on \  -DLAMMPS_EXCEPTIONS=on \  -DCMAKE_INSTALL_PREFIX=... 
make
make install
```
Embed Python code

```python
def simple():
    print("Inside simple function")

simple()
```

Calling existing Python code

```python
simple file my_funcs.py
simple invoke
```
Computes: Call Python function and save result in variable

```python
def factorial(n):
    if n == 1: return 1
    return n*factorial(n-1)

variable fact python factorial
```

- Evaluation of the variable calls the function
Computes: Call Python function and save result in variable

```python
variable n string 10
print "Factorial of $n = \{fact\}"

variable n string 20
print "Factorial of $n = \{fact\}"
```
Pair Python

```
pair_style  python  2.5
pair_coeff  * * py_pot.LJCutMelt lj
```

- (created by Dr. Axel Kohlmeyer)
- for defining simple additive pair potentials in Python
- PYTHONPATH and LAMMPS_POTENTIALS in module search path
- loads class LJCutMelt from user-defined py_pot module
- Python class implements compute_force and compute_energy functions
- many examples, including hybrid usage in examples/python folder
class LJCutMelt(LAMMPSPairPotential):
    def __init__(self):
        super(LJCutMelt, self).__init__()
        # set coeffs: 48*eps*sig**12, 24*eps*sig**6,
        # 4*eps*sig**12, 4*eps*sig**6
        self.units = 'lj'
        self.coeff = {'lj' : {'lj' : (48.0,24.0,4.0,4.0)}}

    def compute_force(self, rsq, itype, jtype):
        coeff = self.coeff[self.pmap[itype]][self.pmap[jtype]]
        r2inv  = 1.0/rsq
        r6inv  = r2inv*r2inv*r2inv
        lj1    = coeff[0]
        lj2    = coeff[1]
        return (r6inv * (lj1*r6inv - lj2))*r2inv
Why do this?
Why do this?

Because Python is Awesome!
Why do this?

But Python is slow...
Performance hit

250 timesteps LJ, 4000 atoms

run time in seconds

C++ only: 1.86 seconds
Python only: 15.13 seconds
Why do this?

- Quick prototyping
- You don’t have to recompile LAMMPS to test it
- You don’t have to work in C++
- And you gain the flexibility / simplicity of the Python language to test new things
Workaround: Create a tabulated version

# use python pair style
pair_style python 2.5
pair_coeff * * py_pot.LJCutMelt lj

# generate tabulated potential from python variant
pair_write 1 1 2000 rsq 0.01 2.5 lj_1_1.table LJ

# switch pair style to tabulated potential
pair_style table linear 2000
pair_coeff 1 1 lj_1_1.table LJ
Performance hit

250 timesteps LJ, 4000 atoms

- C++ only: 1.86 seconds
- Tabulated Python: 1.99 seconds
- Python only: 15.13 seconds
Fix Python

- like any other fix it implements callback functions for certain events in integration loop
- executed every N time steps
- currently limited to `post_force` and `end_of_step`

```
fix 1 all nve
fix 2 all python 50 end_of_step end_of_step_callback
fix 3 all python 50 post_force post_force_callback
```
from lammps import lammps

def end_of_step_callback(lmp):
    L = lammps(ptr=lmp)
    t = L.extract_global("ntimestep", 0)
    print("### END OF STEP ###", t)

def post_force_callback(lmp, v):
    L = lammps(ptr=lmp)
    t = L.extract_global("ntimestep", 0)
    print("### POST_FORCE ###", t)
Python Interfaces

**lammps.lammps**
- uses C-Types
- direct memory access to native C++ data
- provides functions to send and receive data to LAMMPS
- requires knowledge of how LAMMPS works

**lammps.PyLammps**
- higher-level abstraction built on top of original C-Types interface
- manipulation of Python objects
- communication with LAMMPS is hidden from API user
- shorter, more concise Python code
PyLammps

Motivation

- Create a simpler, Python-like interface to LAMMPS
- API should be discoverable (no knowledge of the C++ code necessary)
- IPython notebook integration

Usage

```python
from lammps import PyLammps
L = PyLammps()
```
Commands

LAMMPS Input Script
region box block 0 10 0 5 -0.5 0.5

Original Python Interface
L.command("region box block 0 10 0 5 -0.5 0.5")

PyLammps
L.region("box block", 0, 10, 0, 5, -0.5, 0.5)
Commands - Easier parametrization

Original Python Interface

```python
L.command( 
    "region box block %f %f %f %f %f %f" % 
    (xlo, xhi, ylo, yhi, zlo, zhi) 
)
```

PyLammps

```python
L.region("box block", xlo, xhi, ylo, yhi, zlo, zhi)
```
PyLammps interface Example

Live Demo: python/examples/pylammps/interface_usage.ipynb
Validating a Dihedral potential

Live Demo: python/examples/pylammps/dihedrals/dihedral.ipynb
Summary & Outlook

Ways to call Python from LAMMPS

▶ compute function mapped to variable
▶ invoke function
▶ pairwise potentials (pair python & pair_write)
▶ fix python for end_of_step and post_force

Ways to access/control LAMMPS from Python

▶ lammps.lammps
▶ lammps.PyLammps (Jupyter Notebooks)

Coming soon

▶ fix python/integrate (implement fix nve in Python)
▶ read and write access to atom properties as numpy arrays
References

Documentation


Example Folders

- (Pair Python Fix Python): examples/python
- (PyLammps and Python Interface): python/examples
Questions?