

Comparing Property Predictions Across Interatomic Potentials

Lucas M. Hale

Zachary T. Trautt

Chandler A. Becker

Python-Based Calculation Framework

Forthcoming extension of the NIST
Interatomic Potentials Repository
Project

Property predictions for all available
interatomic potentials

Allows measurements to be:

- Reproducible
- Well documented
- Transparent
- Iterable (automation)
- Comparable
- Archived

The screenshot shows a Jupyter Notebook interface with the title "IP[y]: Notebook" and the subtitle "IPR_DEMO (autosaved)".

1. Initial Setup

Necessary Python Libraries

```
In [1]: import numpy as np
import subprocess
import os
import matplotlib.pyplot as plt
```

Necessary Parameters

These are the parameters that need to be specified to run the code.

- lammps_exe is the directory location for the LAMMPS executable to use.
- working_rd is the directory location for this Notebook and LAMMPS to run from.
- potential is a dictionary containing the necessary parameters associated with running a particular interatomic potential.

```
In [2]: #Specify LAMMPS run command.
lammps_exe = 'C:\\users\\lmh1\\Documents\\lmp_serial.exe'

#Specify working directory.
working_dir = 'C:\\users\\lmh1\\Documents\\IPR_DEMO'

#Specify potential parameters
potential = {}
potential['pair_style'] = 'eam/alloy'
potential['pair_coeff'] = '* * Al99.eam.alloy Al'
potential['mass'] = 26.981539
potential['units'] = 'metal'
potential['atom_style'] = 'atomic'
```

Optional Parameters

These are optional parameters that control how the calculation is performed.

```
In [3]: minimum_alat = 3.5
maximum_alat = 5.0
n_steps_alat = 100
```

2. LAMMPS Script Generation Function

This code generates the underlying LAMMPS script for performing the necessary simulations.

```
In [4]: def script_gen(pair_style, pair_coeff, mass, units, atom_style,
               alat, axes, shift, spacing, size):
    script = '\n'.join([
        'boundary p p p',
        'units %s %s %s' % (units, units, units),
        'atom_style %s %s' % (atom_style, atom_style),
        '',
        'variable alat0 equal %f % alat',
        'lattice fcc S(alat0) %',
        '    origin %f %f %f % (shift[0], shift[1], shift[2]),
        '    spacing %f %f %f % (spacing[0], spacing[1], spacing[2]),
        '    orient x %i %i %i % (axes[0,0], axes[0,1], axes[0,2]),
        '    orient y %i %i %i % (axes[1,0], axes[1,1], axes[1,2]),
        '    orient z %i %i %i % (axes[2,0], axes[2,1], axes[2,2])',
        '    size %f %f %f' % (size, size, size)
    ])
    return script
```

Interatomic Potentials Repository Project

More than 20 available, just for aluminum!

Many more in the literature.

Which one to use...?

Note that elemental potentials taken from alloy descriptions may not work well for the pure species. This is particularly true if the elements were fit for optimized separately. As with all interatomic potentials, please check to make sure that the performance is adequate for your problem.

Al, Al-Co, Al-Co-Ni, Al-Cu, Al-Fe, Al-H, Al-Mg, Al-Mn-Pd, Al-Ni, Al-Pb, Al-Si-Mg-Cu-Fe, Al-Sm, Al-Ti, Nb-Ti-Al, Ni-Al-H

Aluminum (Al)

J.M. Winey, Alison Kubota, and Y.M. Gupta, "A thermodynamic approach to determine accurate potentials for molecular dynamics simulations: thermoelastic response of aluminum," *Modelling Simul. Mater. Sci. Eng.* 17, 055004 (2009). DOI: 10.1088/0965-0393/17/5/055004; *Modelling Simul. Mater. Sci. Eng.* 18, 029801 (2010). DOI: 10.1088/0965-0393/18/2/029801.

Notes: This file was sent by Jonathan Zimmerman (Sandia National Laboratory) and approved for distribution by Michael Winey (Washington State University). It was posted on 16 March 2011.

Format: EAM/alloy (setfl)
File(s): Al_wkg_MSMSE_2009.set

M.I. Mendelev, M.J. Kramer, C.A. Becker, and M. Asta, "Analysis of semi-empirical interatomic potentials appropriate for simulation of crystalline and liquid Al and Cu," *Phil. Mag.* 88, 1723-1750 (2008). DOI: 10.1080/14786430802206482.

Notes: These files were provided by Mikhail Mendelev.

Format: EAM/FS setfl
File(s): Al1.eam.fs

X.W. Zhou, R.A. Johnson, and H.N.G. Wadley, "Misfit-energy-increasing dislocations in vapor-deposited CoFe/NiFe multilayers," *Phys. Rev. B*, 69, 144113 (2004). DOI: 10.1103/PhysRevB.69.144113.

Notes: This file was generated by C.A. Becker from the files sent by X.W. Zhou (Sandia National Laboratory) and posted with his permission. These files can be used to generate alloy potentials for Cu, Ag, Au, Ni, Pd, Pt, Al, Pb, Fe, Mo, Ta, W, Mg, Co, Ti, and Zr by editing EAM.input. However, as addressed in the reference, these potentials were not designed for use with metal compounds. See the Zhou04 page for more information.

Format: EAM/alloy setfl
File(s): Al.set

X.-Y. Liu, F. Ercolessi, and J.B. Adams, "Aluminium interatomic potential from density functional theory calculations with improved stacking fault energy," *Modelling Simul. Mater. Sci. Eng.* 12, 665-670 (2004). DOI: 10.1088/0965-0393/12/4/007.

Notes: NEWAI.txt was obtained from <http://enpub.fulton.asu.edu/cms/potentials/main/main.htm> and posted with the permission of J.B. Adams. Al-LEA.eam.alloy is a version of the same potential which has been formatted for use in LAMMPS ("D" was replaced by "e", "FCC" by "fcc", and "Al" was added on line 3).

Format: EAM
File(s): NEWAI.txt

Format: EAM setfl
File(s): Al-LEA.eam.alloy

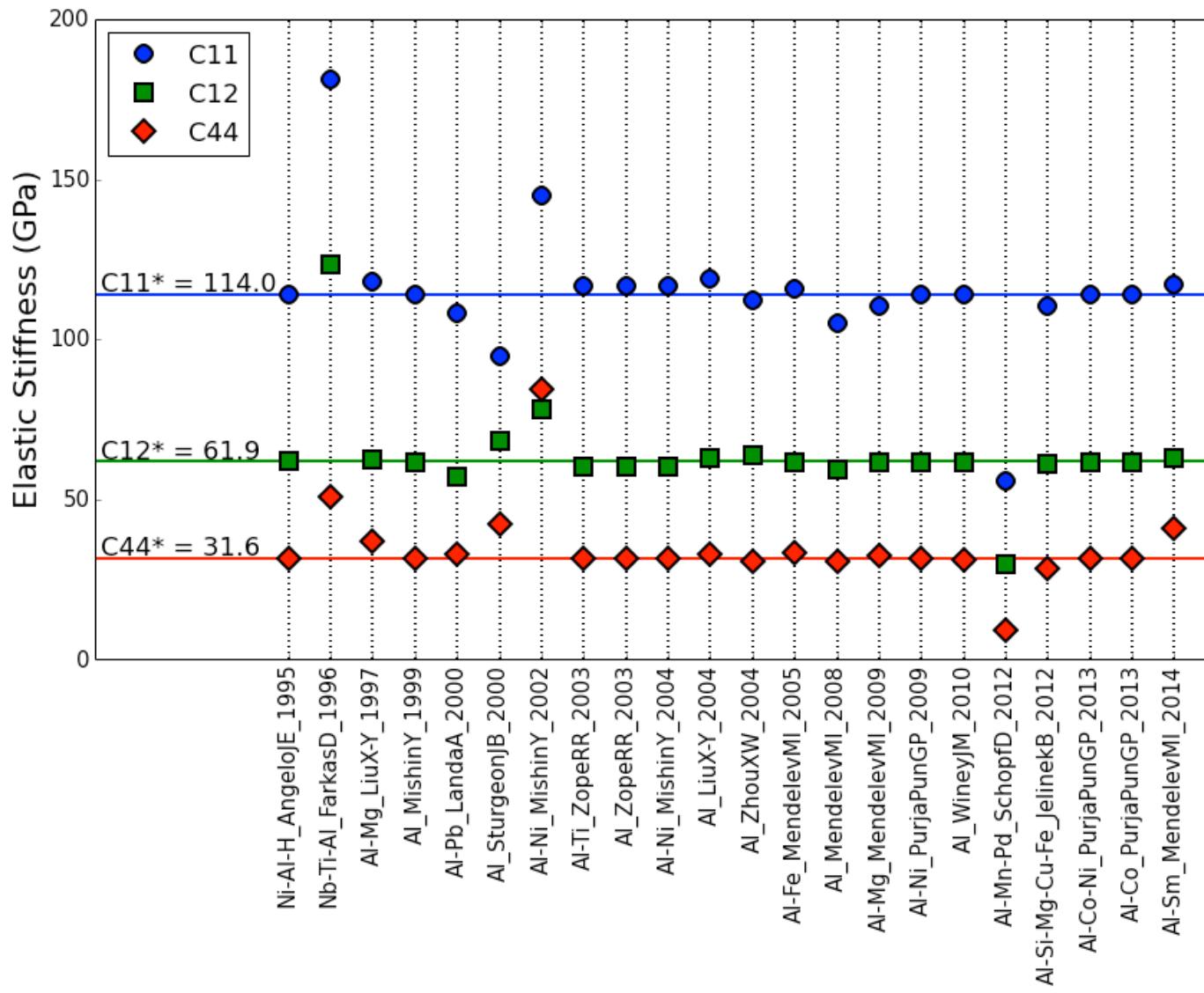
R.R. Zope and Y. Mishin, "Interatomic potentials for atomistic simulations of the Ti-Al system," *Phys. Rev. B* 68, 024102 (2003). DOI: 10.1103/PhysRevB.68.024102.

Notes: This conversion was produced by Chandler Becker on 4 February 2009 from the plt files listed below. This version is compatible with LAMMPS. Validation and usage information can be found in Al03_releaseNotes_1.pdf. If you use this setfl file, please credit the website in addition to the original reference.

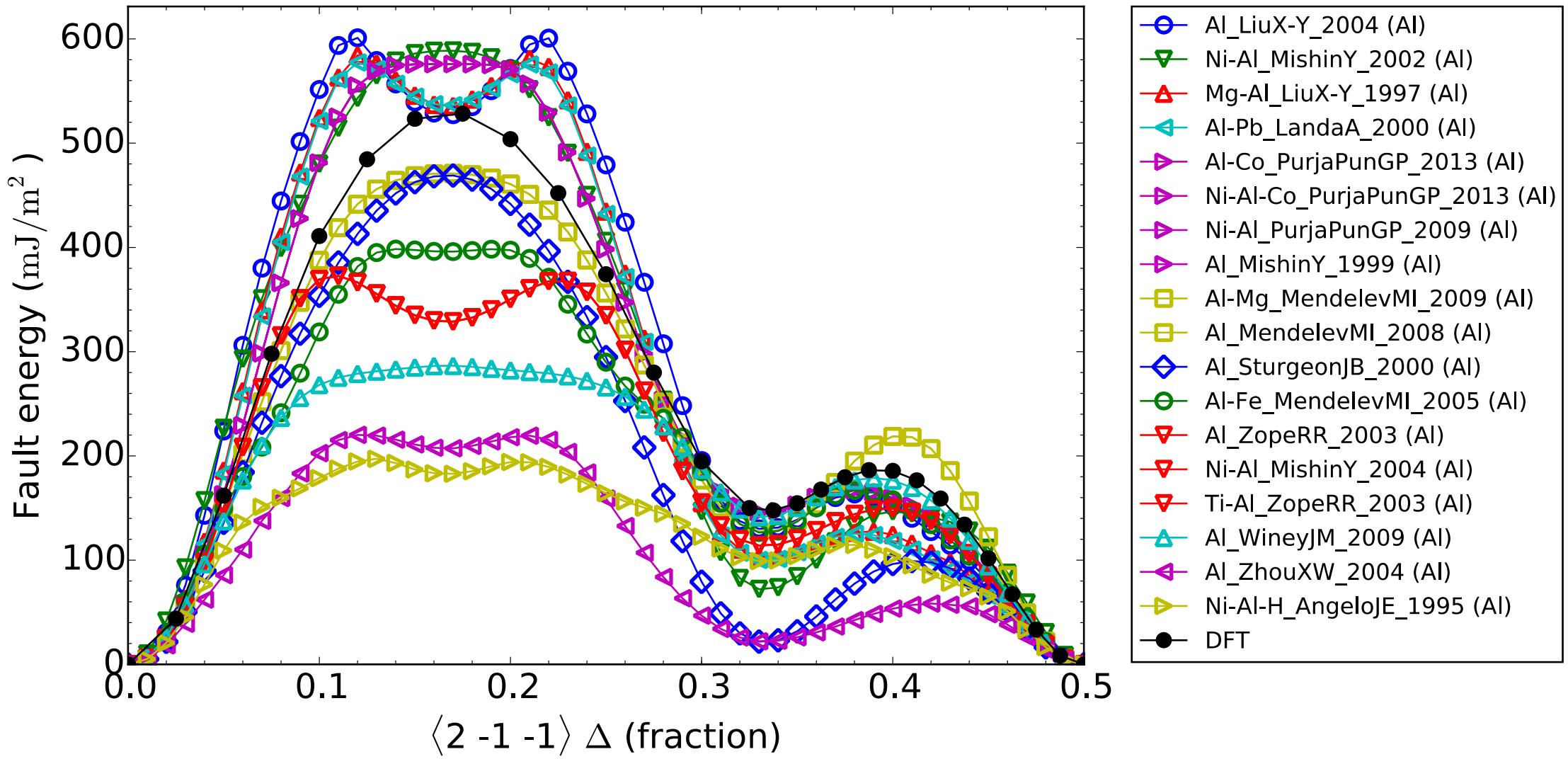
Format: EAM/alloy setfl
File(s): Al03.eam.alloy

Notes: These files were provided by Yuri Mishin.

Property Comparison

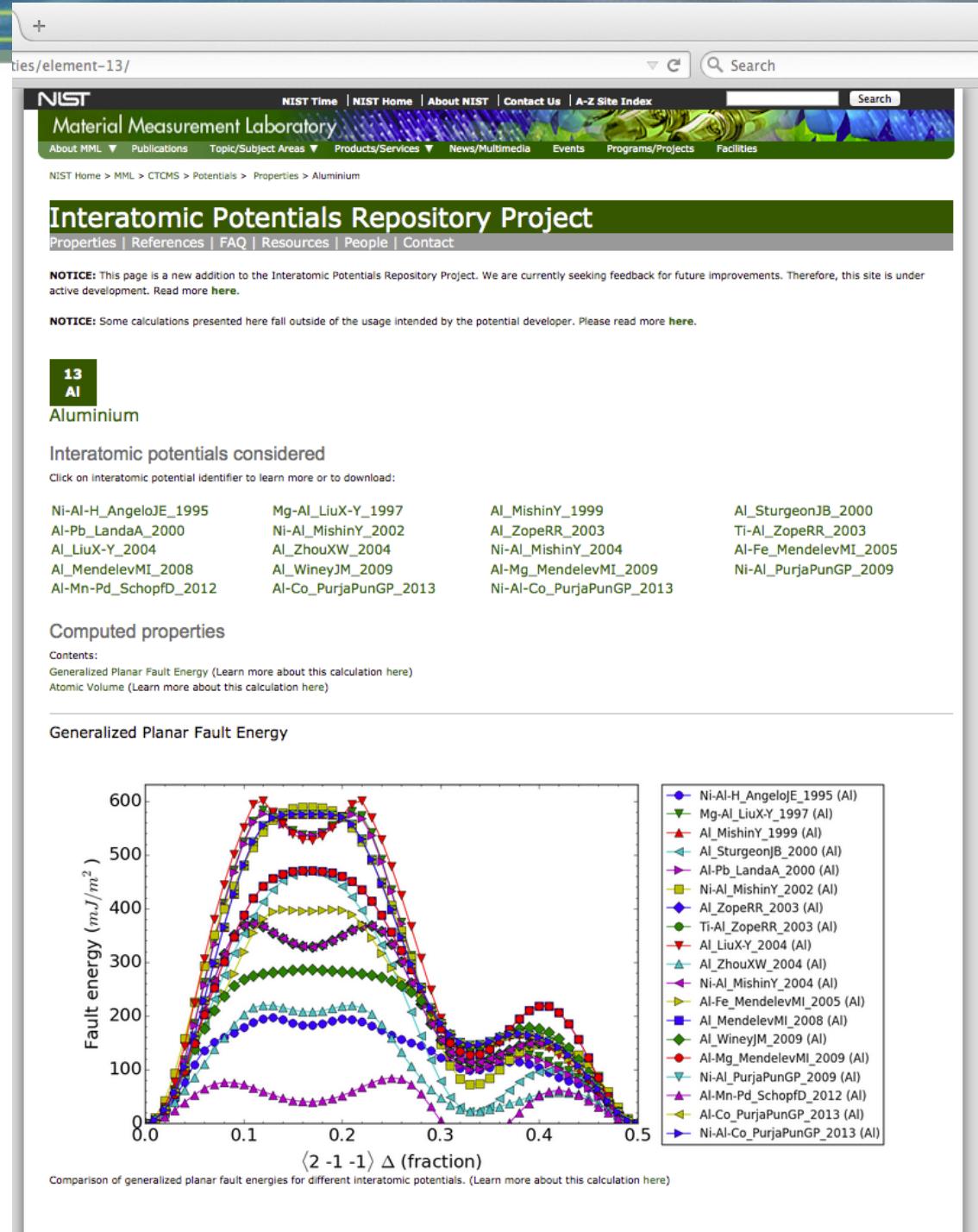


Property Comparison



Updated Website Coming Soon!

- Element-wise comparison pages
- Potential specific breakdowns
- Dynamically searchable results
- All calculation tools used



LAMMPS Input script

Compute energy of fcc system for different sizes and orientations

```
boundary p p p
units metal
atom style atomic
```

Potential Specific Parameters

```
variable alat0 equal 3.25
lattice fcc ${alat0} &
    origin 0.0 0.0 0.0 &
    spacing 1.0 1.0 1.0 &
    orient x 1 0 0 &
    orient y 0 1 0 &
    orient z 0 0 1
```

Simulation Design Parameters

```
region box block 0 3 0 3 0 3
create_box 1 box
create_atoms 1 box
```

```
thermo_style custom step pe
```

```
mass 1 26.981539
pair_style eam/alloy
pair_coeff * * Al99.eam.alloy Al
```

Potential Specific Parameters

```
run 0
```

Python script_gen()

```
def script_gen(pair_style, pair_coeff, mass, units, atom_style,
               alat, axes, shift, spacing, size):

    script = '\n'.join(['boundary p p p',
                       'units %s' % units,
                       'atom_style %s' % atom_style,
                       '',
                       'variable alat0 equal %f' % alat,
                       'lattice fcc ${alat0} &',
                       '  origin %f %f %f &' % (shift[0], shift[1], shift[2]),
                       '  spacing %f %f %f &' % (spacing[0], spacing[1], spacing[2]),
                       '  orient x %i %i %i &' % (axes[0,0], axes[0,1], axes[0,2]),
                       '  orient y %i %i %i &' % (axes[1,0], axes[1,1], axes[1,2]),
                       '  orient z %i %i %i' % (axes[2,0], axes[2,1], axes[2,2]),
                       '',
                       'region box block %i %i %i %i %i %i' % (size[0,0], size[0,1],
                                                               size[1,0], size[1,1],
                                                               size[2,0], size[2,1]),
                       'create_box 1 box',
                       'create_atoms 1 box',
                       '',
                       'thermo_style custom step pe',
                       '',
                       'mass 1 %f' % mass,
                       'pair_style %s' % pair_style,
                       'pair_coeff %s' % pair_coeff,
                       '',
                       'run 0'])

    return script
```

Python Calculation Function

```
def e_vs_a(lammps_exe, potential, amin, amax, asteps):
    avals = []
    evals = []

    #Read potential specific information
    pair_style = potential['pair_style']
    pair_coeff = potential['pair_coeff']
    mass =      potential['mass']
    units =     potential['units']
    atom_style = potential['atom_style']

    #Set constant simulation information
    axes = np.array([ [1, 0, 0], [0, 1, 0], [0, 0, 1] ])
    shift = np.array([0.0, 0.0, 0.0])
    spacing = np.array([1.0, 1.0, 1.0])
    size = np.array([ [0, 3], [0, 3], [0, 3] ])
```

Parameter Setup

```
#Loop over range of lattice parameters
for i in xrange(asteps):
    alat = amin + i * (amax - amin) / (asteps - 1)

    #Generate LAMMPS script
    with open('alat.in','w') as script:
        script.write(script_gen(pair_style, pair_coeff, mass, units, atom_style,
                               alat, axes, shift, spacing, size))

    #Run LAMMPS
    output = subprocess.check_output(lammps_exe+' < alat.in',shell=True)

    #Distil results
    energy = extract_results(output)
    avals.append(alat)
    evals.append(energy)

return avals, evals
```

Run simulation(s)

Process results

Demonstration IPython Notebooks

Mimics Mathematica Notebooks

Runs Python

Allows for complete documentation

- Comments describing calculation method
- Input parameters used
- Underlying LAMMPS script
- Code for calculation
- Results

Easy to learn, reproduce, and adapt

The screenshot shows an IPython Notebook window titled "IP[y]: Notebook IPR_DEMO (autosaved)". The URL in the address bar is "127.0.0.1:8888/notebooks/IPR_DEMO.ipynb". The notebook content includes:

- FCC Lattice Parameter Identifier**
Lucas M. Hale, lucas.hale@nist.gov, Materials Science and Engineering Division, NIST.
Chandler A. Becker, chandler.becker@nist.gov, Materials Science and Engineering Division, NIST.
Zachary T. Trautt, zachary.trautt@nist.gov, Materials Measurement Science Division, NIST.
Version: 2015-07-17
[Disclaimers](#)
- Software**
This notebook was tested with:
 - LAMMPS (Version 2015-03-27)
 - Python (Version 2.7.6)
 - IPython (Version 2.0.0)
- Introduction**
This is an example of how a Notebook is formatted.
- 1. Initial Setup**
Necessary Python Libraries
In [1]:

```
import numpy as np
import subprocess
import os
import matplotlib.pyplot as plt
```
- Necessary Parameters**
These are the parameters that need to be specified to run the code.
 - lammps_exe is the directory location for the LAMMPS executable to use.
 - working_dr is the directory location for this Notebook and LAMMPS to run from.
 - potential is a dictionary containing the necessary parameters associated with running a particular interatomic potential.
- In [2]:**

```
#Specify LAMMPS run command.
lammps_exe = 'C:\\users\\lmh1\\Documents\\lmp_serial.exe'

#Specify working directory.
working_dr = 'C:\\users\\lmh1\\Documents\\IPR_DEMO'

#Specify potential parameters
potential = {}
```

Demonstration IPython Notebooks

Mimics Mathematica Notebooks

Runs Python

Allows for complete documentation

- Comments describing calculation method
- Input parameters used
- Underlying LAMMPS script
- Code for calculation
- Results

Easy to learn, reproduce, and adapt

The screenshot shows an IPython Notebook window titled "IP[y]: Notebook IPR_DEMO (autosaved)". The browser tab is "IPy IPR_DEMO". The notebook contains several sections and code cells:

- 1. Initial Setup**
 - Necessary Python Libraries**

```
In [1]: import numpy as np
import subprocess
import os
import matplotlib.pyplot as plt
```
 - Necessary Parameters**

These are the parameters that need to be specified to run the code.

 - lammps_exe is the directory location for the LAMMPS executable to use.
 - working_rd is the directory location for this Notebook and LAMMPS to run from.
 - potential is a dictionary containing the necessary parameters associated with running a particular interatomic potential.

```
In [2]: #Specify LAMMPS run command.
lammps_exe = 'C:\\users\\lmh1\\Documents\\lmp_serial.exe'

#Specify working directory.
working_dr = 'C:\\users\\lmh1\\Documents\\IPR_DEMO'

#Specify potential parameters
potential = {}
potential['pair_style'] = 'eam/alloy'
potential['pair_coeff'] = '1 * Al99.eam.alloy Al'
potential['mass'] = 26.981539
potential['units'] = 'metal'
potential['atom_style'] = 'atomic'
```

 - Optional Parameters**

These are optional parameters that control how the calculation is performed.

```
In [3]: minimum_alet = 3.5
maximum_alet = 5.0
n_steps_alet = 100
```- 2. LAMMPS Script Generation Function**

This code generates the underlying LAMMPS script for performing the necessary simulations.

```
In [4]: def script_gen(pair_style, pair_coeff, mass, units, atom_style,
                 alet, axes, shift, spacing, size):

    script = '\n'.join(['boundary p p p',
                      'units $' + units,
                      'atom_style $' + atom_style,
                      '',
                      'variable alet0 equal $f' + alet,
                      'lattice fcc $[alet0] $',
                      'origin $f $f $f $ % (shift[0], shift[1], shift[2]),
                      spacing $f $f $f $ % (spacing[0], spacing[1], spacing[2]),
                      orient x $1 $1 $1 $ % (axes[0,0], axes[0,1], axes[0,2]),
                      orient y $1 $1 $1 $ % (axes[1,0], axes[1,1], axes[1,2]),
```

Demonstration IPython Notebooks

Mimics Mathematica Notebooks

Runs Python

Allows for complete documentation

- Comments describing calculation method
- Input parameters used
- Underlying LAMMPS script
- Code for calculation
- Results

Easy to learn, reproduce, and adapt

The screenshot shows an IPython Notebook interface with two code cells displayed.

Cell 4: 2. LAMMPS Script Generation Function

```
In [4]: def script_gen(pair_style, pair_coeff, mass, units, atom_style, alat, axes, shift, spacing, size):
    script = '\n'.join(['boundary p p p',
                       'units $' + units,
                       'atom_style $' + atom_style,
                       '',
                       'variable alat0 equal $f' + alat,
                       'lattice fcc $alat0 &',
                       '    origin $f $f $f & (shift[0], shift[1], shift[2]),
                       '    spacing $f $f $f & (spacing[0], spacing[1], spacing[2]),
                       '    orient x $i $i $i & (axes[0,0], axes[0,1], axes[0,2]),
                       '    orient y $i $i $i & (axes[1,0], axes[1,1], axes[1,2]),
                       '    orient z $i $i $i & (axes[2,0], axes[2,1], axes[2,2]),
                       '',
                       'region box block $i $i $i $i $i & (size[0,0], size[0,1],
                       '                                         size[1,0], size[1,1],
                       '                                         size[2,0], size[2,1]),
                       'create_box 1 box',
                       'create_atoms 1 box',
                       '',
                       'thermo_style custom step pe',
                       '',
                       'mass 1 %f' + mass,
                       'pair_style $s' + pair_style,
                       'pair_coeff $s' + pair_coeff,
                       '',
                       'run 0'])
    return script
```

Cell 5: 3. Python Calculation Function

```
In [5]: #Measure energy as a function of lattice parameter
def e_vs_a(lammps_exe, potential, amin, amax, asteps):
    avals = []
    evals = []

    #Read potential specific information
    pair_style = potential['pair_style']
    pair_coeff = potential['pair_coeff']
    mass = potential['mass']
    units = potential['units']
    atom_style = potential['atom_style']

    #Set constant simulation information
    axes = np.array([1, 0, 0], [0, 1, 0], [0, 0, 1])
    shift = np.array([0.0, 0.0, 0.0])
    spacing = np.array([1.0, 1.0, 1.0])
    size = np.array([0, 3], [0, 3], [0, 3])

    #Loop over range of lattice parameters
    for i in xrange(asteps):
        alat = amin + i * (amax - amin) / (asteps - 1)
```

Demonstration IPython Notebooks

Mimics Mathematica Notebooks

Runs Python

Allows for complete documentation

- Comments describing calculation method
- Input parameters used
- Underlying LAMMPS script
- Code for calculation
- Results

Easy to learn, reproduce, and adapt

The screenshot shows an IPython Notebook interface with two code cells displayed.

Cell 5: Python Calculation Function

```
#Measure energy as a function of lattice parameter
def e_vs_a(lammps_exe, potential, amin, amax, asteps):
    avals = []
    evals = []

    #Read potential specific information
    pair_style = potential['pair_style']
    pair_coeff = potential['pair_coeff']
    mass = potential['mass']
    units = potential['units']
    atom_style = potential['atom_style']

    #Set constant simulation information
    axes = np.array([ [1, 0, 0], [0, 1, 0], [0, 0, 1] ])
    shift = np.array([0.0, 0.0, 0.0])
    spacing = np.array([1.0, 1.0, 1.0])
    size = np.array([ 3, 3, 3 ])

    #Loop over range of lattice parameters
    for i in xrange(asteps):
        alat = amin + i * (amax - amin) / (asteps - 1)

        #Generate LAMMPS script
        with open('alat.in', 'w') as script:
            script.write(script_gen(pair_style, pair_coeff, mass, units, atom_style,
                                   alat, axes, shift, spacing, size))

        #Run LAMMPS
        output = subprocess.check_output(lammps_exe+' < alat.in', shell=True)

        #Distil results
        energy = extract_results(output)
        avals.append(alat)
        evals.append(energy)

    return avals, evals
```

Cell 6: Energy Extraction Function

```
#Take LAMMPS screen output and parse out the thermo data into a list
def extract_results(output):
    thermo = False
    first = True
    thermolist = []

    lines = output.split('\n')
    for line in lines:
        terms = line.split()

        #If the line has terms
        if len(terms)>0:
            #If the line starts with Step, then set thermo
            if terms[0] == 'Step':
                thermo = True
            #Make the first line of the returned array the thermo data headers
            if first:
```

Demonstration IPython Notebooks

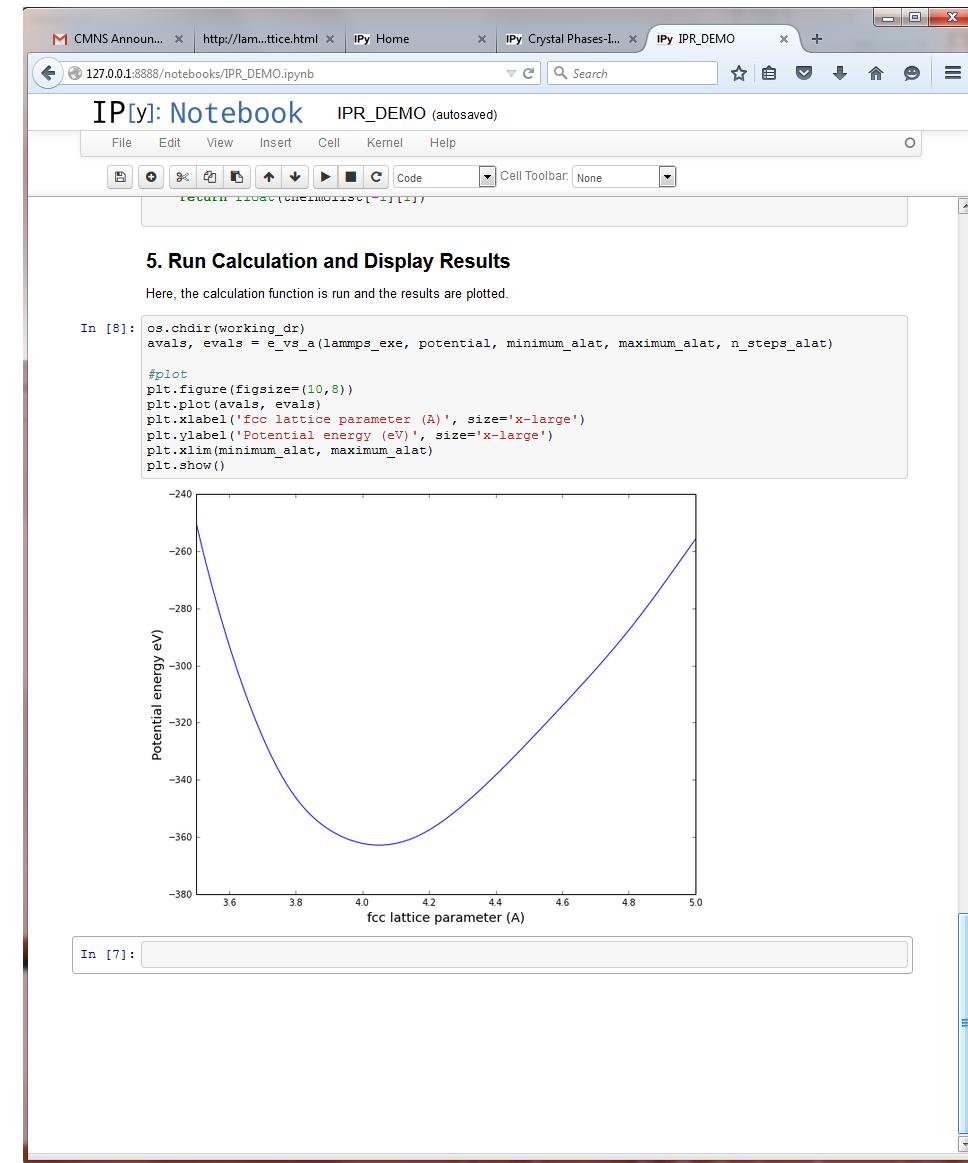
Mimics Mathematica Notebooks

Runs Python

Allows for complete documentation

- Comments describing calculation method
- Input parameters used
- Underlying LAMMPS script
- Code for calculation
- Results

Easy to learn, reproduce, and adapt

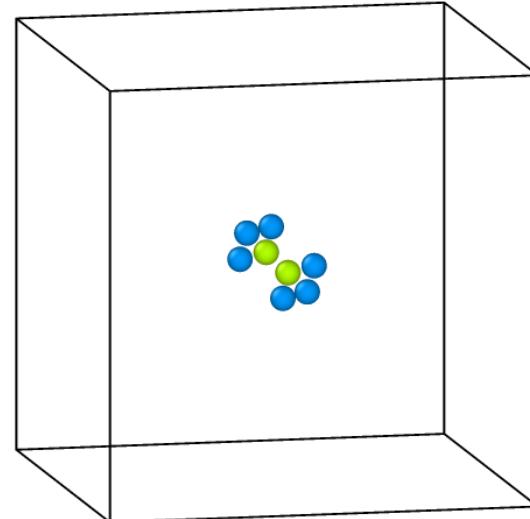


High-Throughput Scripts

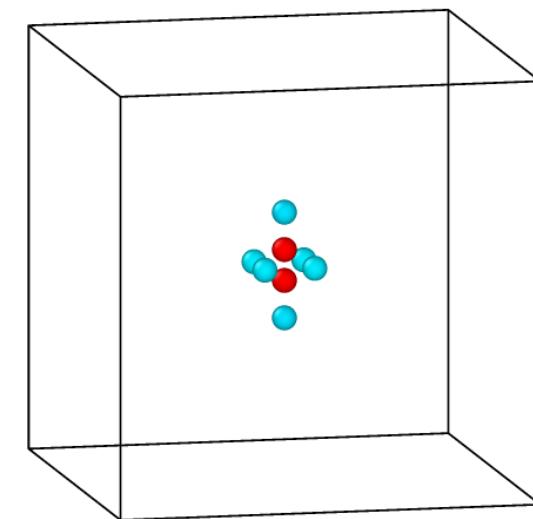
1. Each calculation function should be independent and isolated
 - Serial and parallel looping
 - Cluster distribution
2. Python tools for easy creation and manipulation of atomic systems
3. Data models for handling, representing, and archiving the input and output data

IPRP Python Package

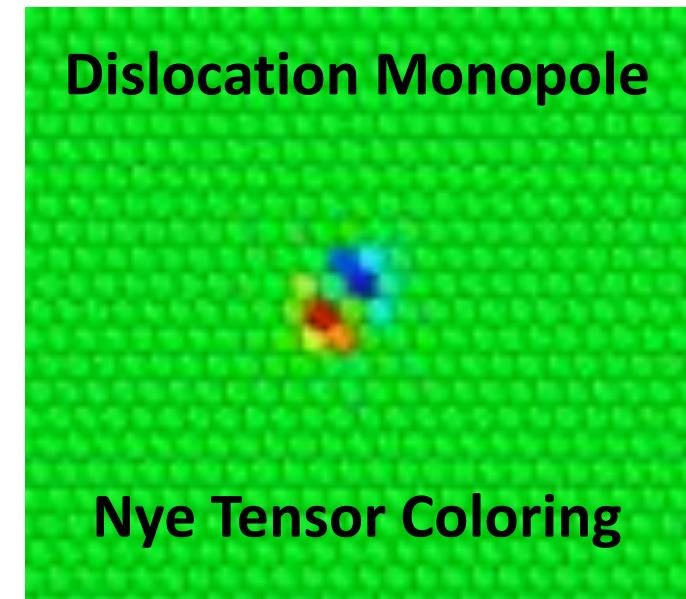
- Main iprp
 - Classes representing atoms, atom types and systems
 - Functions for manipulating systems (i.e. add defects)
- Submodule iprp.lammps
 - Converters for LAMMPS <-> Python system information
 - Basic LAMMPS script generators
- Submodule iprp.models
 - XML <-> json <-> Python conversions
 - Classes associated with specific input data models



[111] interstitial



[100] interstitial



LAMMPS Implementation Data Model

Read file

Supply elements list

Get run parameters

Seamlessly switch potentials!

Why pick only one potential?

```
In [1]: import iprp
In [2]: import iprp.lammps
In [3]: pot = iprp.lammps.Potential('C:/lmp-U-Mo-Xe_SmirnovaD_2013.json')
In [4]: print pot.coeff(['U','U','Xe'])
pair_style eam/alloy
pair_coeff * * U_Mo_Xe.2013.eam.alloy U U Xe
In [5]:
```

```
{
    "interatomicPotentialImplementationLAMMPS": {
        "potentialID": {
            "descriptionIdentifier": "U-Mo-Xe_SmirnovaD_2013"
        },
        "units": "metal",
        "atom_style": "atomic",
        "atom": [
            {
                "element": "U",
                "mass": "238.02891"
            },
            {
                "element": "Mo",
                "mass": "95.96"
            },
            {
                "element": "Xe",
                "mass": "131.293"
            }
        ],
        "pair_style": {
            "type": "eam/alloy"
        },
        "pair_coeff": [
            {
                "term": [
                    {
                        "file": "U_Mo_Xe.2013.eam.alloy"
                    },
                    {
                        "symbols": "True"
                    }
                ]
            }
        ]
    }
}
```

LAMMPS Implementation Data Model

```
{"interatomicPotentialImplementationLAMMPS": {  
    "potentialID":  
        {"descriptionIdentifier": "Al-Si-Mg-Cu-Fe_JelinekB_2012"},  
    "units": "metal",  
    "atom_style": "atomic",  
    "atom": [  
        {"element": "Al", "symbol": "AlS"},  
        {"element": "Si", "symbol": "SiS"},  
        {"element": "Mg", "symbol": "MgS"},  
        {"element": "Cu", "symbol": "CuS"},  
        {"element": "Fe", "symbol": "FeS"}  
    "pair_style": {"type": "meam"},  
    "pair_coeff": {  
        "term": [  
            {"file": "Jelinek_2012_meamf"},  
            {"symbolsList": True},  
            {"file": "Jelinek_2012_meam.alsimgcufe"},  
            {"symbols": True}]}  
}
```

LAMMPS Implementation Data Model

```
{"interatomicPotentialImplementationLAMMPS": {  
    "potentialID":  
        {"descriptionIdentifier": "MO_751354403791_001"},  
    "units": "metal",  
    "atom_style": "atomic",  
    "atom": [  
        {"element": "Al"},  
        {"element": "Ni"}  
    "pair_style": {  
        "type": "kim",  
        "term":  
            {"option": "KIMvirial MO_751354403791_001"}  
    "pair_coeff": {"term": {"symbols": "True"} }  
}
```

LAMMPS Implementation Data Model

```
{"interatomicPotentialImplementationLAMMPS": {  
    "potentialID": {  
        "descriptionIdentifier": "O-Cu-N-C-H-Ti-Zn_LiangT_2013"},  
    "units": "metal",  
    "atom_style": "charge",  
    "atom": [  
        {"element": "O"}, {"element": "Cu"},  
        {"element": "N"}, {"element": "C"},  
        {"element": "H"}, {"element": "Ti"},  
        {"element": "Zn"}  
    "pair_style": {"type": "comb"},  
    "pair_coeff": {  
        "term": [{"file": "ffield.comb"}, {"symbols": "True"}]  
    "supplement":{  
        "term": {"option": "fix qeq all qeq/comb 1 0.0001"}  
}
```

“Idea” of an
interatomic model

Unique
Identifier
UID#

Interatomic model
metadata with human
readable nickname

Before publication
Ni—MendelevMI—
2010

After publication
Ni—MendelevMI—
2012

Implementations of
interatomic model with
implementation UID#

EAM

OpenKIM

EAM

Artifacts located at a
repository

Setfl file

Setfl file
(updated
comments)

“Idea” of an
interatomic model

Unique
Identifier
UID#

Interatomic model
metadata with human
readable nickname

Before publication
Ni—MendelevMI—
2010

After publication
Ni—MendelevMI—
2012

Implementations of
interatomic model with
implementation UID#

EAM

OpenKIM

EAM

Artifacts located at a
repository

Setfl file

Setfl file
(updated
comments)

Want To Help?

- Scripts for basic materials properties
- Beta testers for iprp package and Python scripts
- Community involvement for standardized interatomic model metadata and citations
- lucas.hale@nist.gov