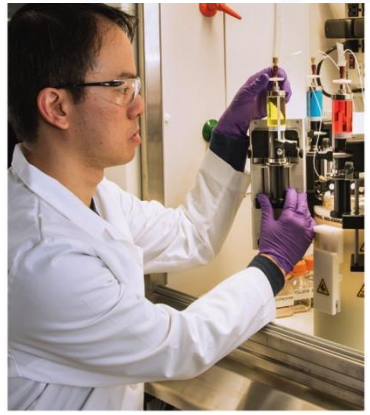
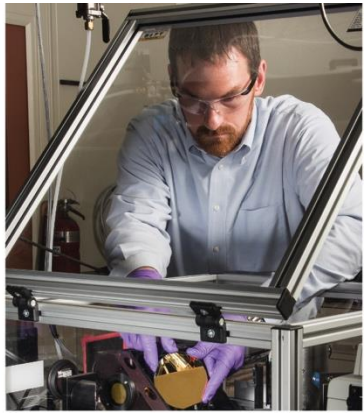




Interatomic Potentials Repository Tools and Resources

Lucas M. Hale

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

IPR

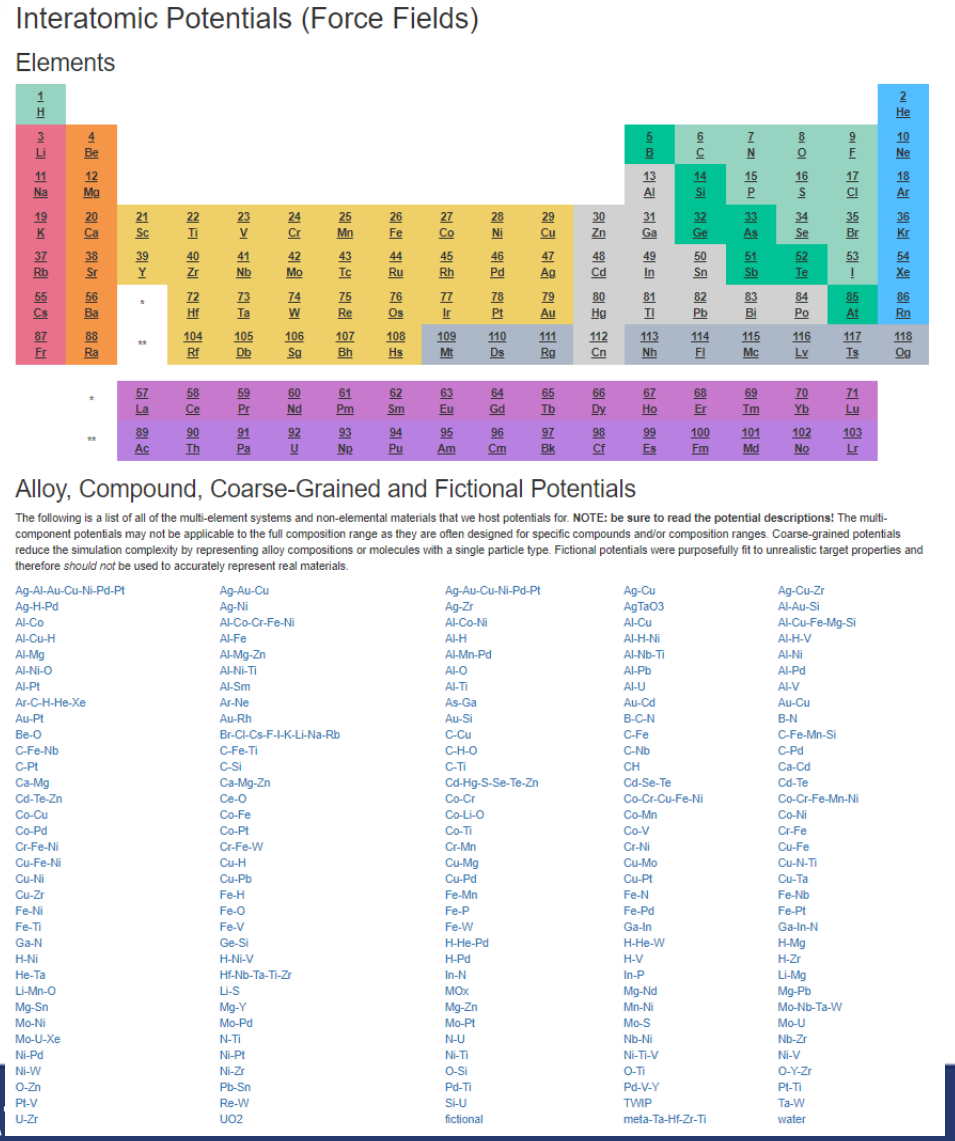
MATERIAL MEASUREMENT LABORATORY

INTERATOMIC POTENTIALS REPOSITORY

HTTPS://WWW.CTCMS.NIST.GOV/POTENTIALS/

Search and explore existing potentials

- Predominately for hard materials
- Parameter files with known provenance
- Additional files (documentation, patches, codes, validation data, …)
- Links to externally hosted potentials (OpenKIM)
- Citation information
- Notes on usage
- List of related models
- Descriptions of different versions
- Property calculations



Interatomic Potentials (Force Fields)

Elements

1 H																	2 He
3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
55 Cs	56 Ba	*	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt	110 Ds	111 Rg	112 Cn	113 Nh	114 Fl	115 Mc	116 Lv	117 Ts	118 Og

*	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
**	89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

Alloy, Compound, Coarse-Grained and Fictional Potentials

The following is a list of all of the multi-element systems and non-elemental materials that we host potentials for. **NOTE: be sure to read the potential descriptions!** The multi-component potentials may not be applicable to the full composition range as they are often designed for specific compounds and/or composition ranges. Coarse-grained potentials reduce the simulation complexity by representing alloy compositions or molecules with a single particle type. Fictional potentials were purposefully fit to unrealistic target properties and therefore *should not* be used to accurately represent real materials.

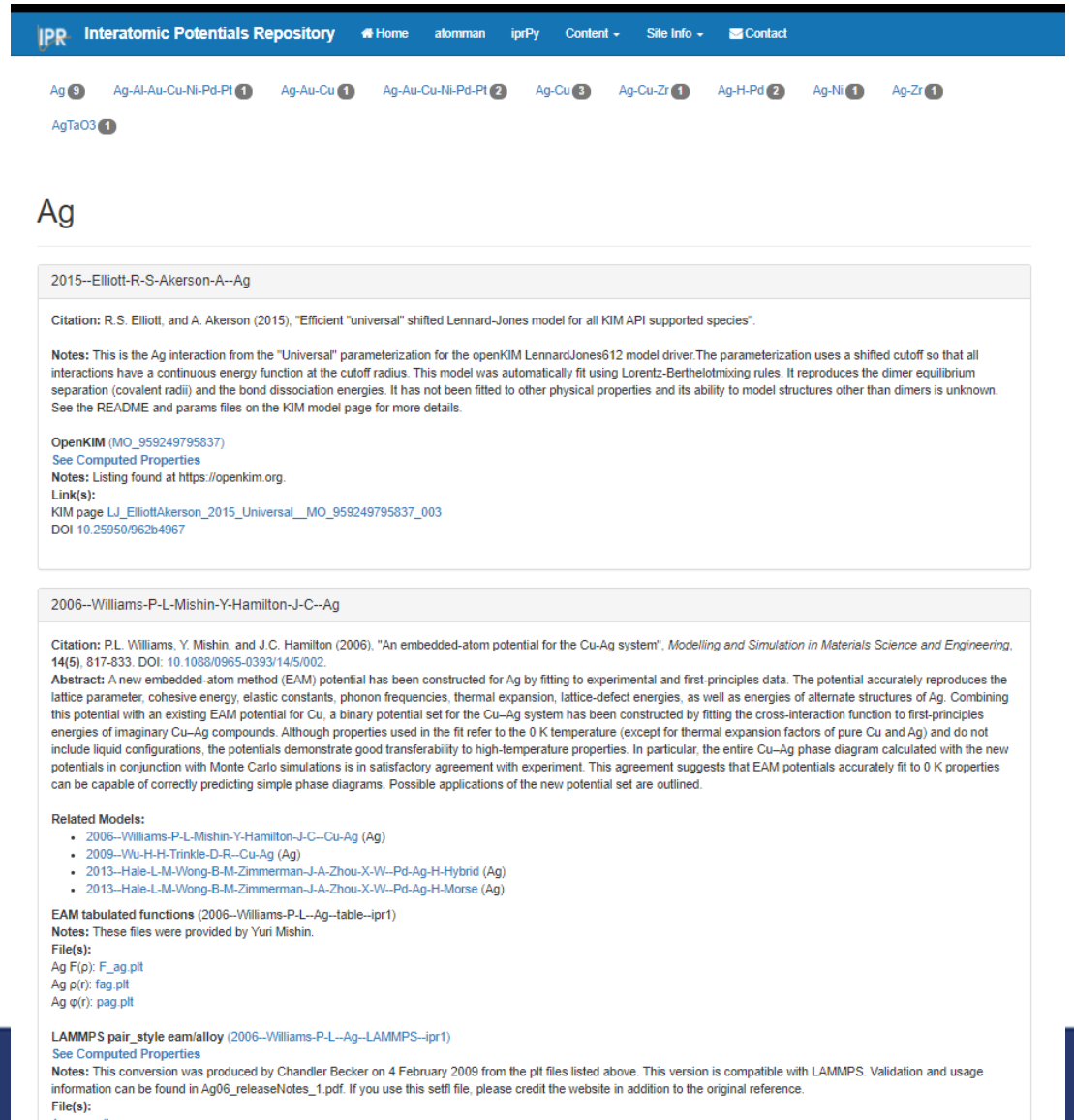
Ag-Al-Au-Cu-Ni-Pd-Pt	Ag-Au-Cu	Ag-Au-Cu-Ni-Pd-Pt	Ag-Cu	Ag-Cu-Zr
Ag-H-Pd	Ag-Ni	Ag-Zr	AgTaO3	Al-Au-Si
Al-Co	Al-Co-Cr-Fe-Ni	Al-Co-Ni	Al-Cu	Al-Cu-Fe-Mg-Si
Al-Cu-H	Al-Fe	Al-H	Al-H-Ni	Al-H-V
Al-Mg	Al-Mg-Zn	Al-Mn-Pd	Al-Nb-Ti	Al-Ni
Al-Ni-O	Al-Ni-Ti	Al-O	Al-Pb	Al-Pd
Al-Pt	Al-Sm	Al-Ti	Al-U	Al-V
Ar-C-H-He-Xe	Ar-Ne	As-Ga	Au-Cd	Au-Cu
Au-Pt	Au-Rh	Au-Si	B-C-N	B-N
Be-O	Br-Cl-Cs-F-I-K-Li-Na-Rb	C-Cu	C-Fe	C-Fe-Mn-Si
C-Fe-Nb	C-Fe-Ti	C-H-O	C-Nb	C-Pd
C-Pt	C-Si	C-Ti	CH	Ca-Cd
Ca-Mg	Ca-Mg-Zn	Cd-Hg-S-Se-Te-Zn	Cd-Se-Te	Cd-Te
Cd-Te-Zn	Ce-O	Co-Cr	Co-Cr-Cu-Fe-Ni	Co-Cr-Fe-Mn-Ni
Co-Cu	Co-Fe	Co-Li-O	Co-Mn	Co-Ni
Co-Pd	Co-Pt	Co-Ti	Co-V	Cr-Fe
Cr-Fe-Ni	Cr-Fe-W	Cr-Mn	Cr-Ni	Cu-Fe
Cu-Fe-Ni	Cu-H	Cu-Mg	Cu-Mo	Cu-N-Ti
Cu-Ni	Cu-Pb	Cu-Pd	Cu-Pt	Cu-Ta
Cu-Zr	Fe-H	Fe-Mn	Fe-N	Fe-Nb
Fe-Ni	Fe-O	Fe-P	Fe-Pd	Fe-Pt
Fe-Ti	Fe-V	Fe-W	Ga-In	Ga-In-N
Ga-N	Ge-Si	H-He-Pd	H-He-W	H-Mg
H-Ni	H-Ni-V	H-Pd	H-V	H-Zr
He-Ta	Hf-Nb-Ta-Ti-Zr	In-N	In-P	Li-Mg
Li-Mn-O	Li-S	MOx	Mg-Nd	Mg-Pb
Mg-Sn	Mg-Y	Mg-Zn	Mn-Ni	Mo-Nb-Ta-W
Mo-Ni	Mo-Pd	Mo-Pt	Mo-S	Mo-U
Mo-U-Xe	N-Ti	N-U	Nb-Ni	Nb-Zr
Ni-Pd	Ni-Pt	Ni-Ti	Ni-Ti-V	Ni-V
Ni-W	Ni-Zr	O-Si	O-Ti	O-Y-Zr
O-Zn	Pb-Sn	Pd-Ti	Pd-V-Y	Pt-Ti
Pt-V	Re-W	Si-U	TWIP	Ta-W
U-Zr	UO2	fictional	meta-Ta-Hf-Zr-Ti	water

INTERATOMIC POTENTIALS REPOSITORY

HTTPS://WWW.CTCMS.NIST.GOV/POTENTIALS/

Search and explore existing potentials

- Predominately for hard materials
- Parameter files with known provenance
- Additional files (documentation, patches, codes, validation data, …)
- Links to externally hosted potentials (OpenKIM)
- Citation information
- Notes on usage
- List of related models
- Descriptions of different versions
- Property calculations



CRYSTAL AND DEFECT CALCULATIONS

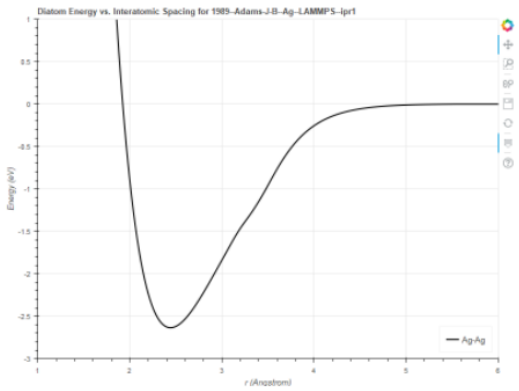
View computed properties

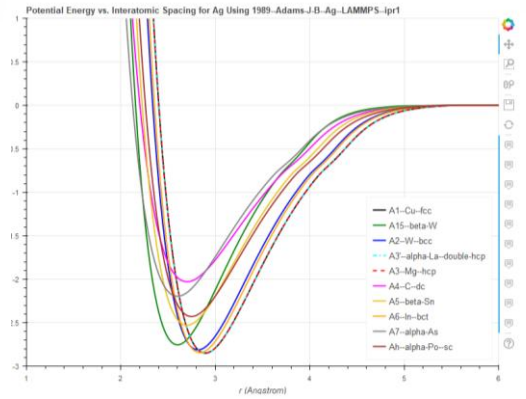
Download raw data

Descriptions of the methods

Links to the code used

New properties in progress…





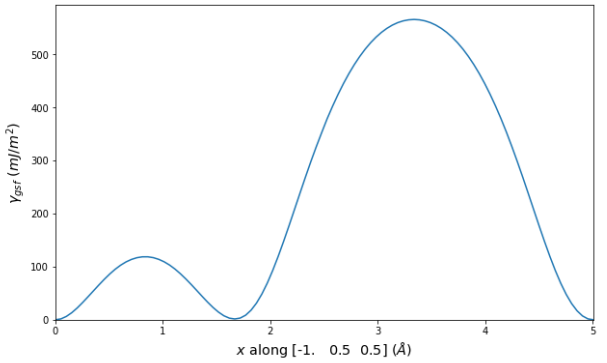
Cij in GPa:

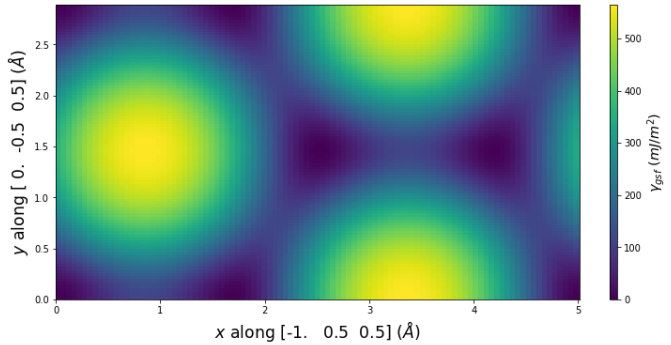
129.069	91.074	91.074	0.0	0.0	-0.0
91.074	129.069	91.074	-0.0	0.0	0.0
91.074	91.074	129.069	0.0	0.0	0.0
0.0	0.0	0.0	56.743	0.0	0.0
0.0	0.0	0.0	0.0	56.743	0.0
0.0	0.0	0.0	-0.0	0.0	56.743

prototype	method	E_{coh} (eV/atom)	E_{pot} (eV/atom)	a_0 (Å)	b_0 (Å)	c_0 (Å)	α (degrees)	β (degrees)	γ (degrees)
A1--Cu--fcc	dynamic	-2.85	-2.85	4.09	4.09	4.09	90.0	90.0	90.0
A3'--alpha-La--double-hcp	dynamic	-2.8498	-2.8498	2.8923	2.8923	9.4424	90.0	90.0	120.0
mp-989737	dynamic	-2.8498	-2.8498	2.8923	2.8923	21.2434	90.0	90.0	120.0
A3--Mg--hcp	dynamic	-2.8497	-2.8497	2.8925	2.8925	4.7198	90.0	90.0	120.0
A2--W--bcc	box	-2.8133	-2.8133	3.2517	3.2517	3.2517	90.0	90.0	90.0
oqmd-1214858	dynamic	-2.793	-2.793	7.0562	7.0562	7.0562	90.0	90.0	90.0
oqmd-1214858	box	-2.7926	-2.7926	7.0545	7.0545	7.0545	90.0	90.0	90.0
oqmd-1214769	dynamic	-2.7785	-2.7785	10.0517	10.0517	10.0517	90.0	90.0	90.0
oqmd-1214769	box	-2.7784	-2.7784	10.0518	10.0518	10.0518	90.0	90.0	90.0
A15--beta-W	static	-2.7543	-2.7543	5.1989	5.1989	5.1989	90.0	90.0	90.0
oqmd-1214680	box	-2.6325	-2.6325	4.0061	9.7678	3.9984	90.0	90.0	90.0
A5--beta-Sn	static	-2.5425	-2.5425	5.2833	5.2833	2.773	90.0	90.0	90.0
Ah--alpha-Po--sc	static	-2.4289	-2.4289	2.7463	2.7463	2.7463	90.0	90.0	90.0
oqmd-1215927	static	-2.1819	-2.1819	4.6482	4.6482	5.2122	90.0	90.0	120.0
A4--C--dc	static	-2.0294	-2.0294	6.2375	6.2375	6.2375	90.0	90.0	90.0

Point Defect	E_f (eV)	p_{11} (eV)	p_{22} (eV)	p_{33} (eV)	p_{12} (eV)	p_{13} (eV)	p_{23} (eV)
vacancy	0.968	-1.967	-1.967	-1.967	0.0	0.0	-0.0
1nn divacancy	1.805	-3.971	-3.717	-3.717	0.0	-0.0	0.039
2nn divacancy	1.935	-4.125	-4.125	-4.067	-0.0	0.0	-0.0
100 dumbbell	3.261	22.74	22.74	21.946	0.0	0.0	0.0
crowdion interstitial	3.5	21.672	21.672	26.088	11.277	-0.0	-0.0
110 dumbbell	3.5	26.078	21.652	21.652	-0.0	-0.0	11.275
tetrahedral interstitial	3.808	23.448	23.448	23.448	-0.0	-0.0	-0.0

Surface	γ_{fs} (mJ/m^2)
(111)	618.45
(332)	686.81
(322)	693.97
(100)	703.27
(221)	716.94
(211)	731.27
(331)	741.44
(311)	758.9
(110)	765.25
(321)	770.77
(310)	792.22
(320)	800.28
(210)	807.88





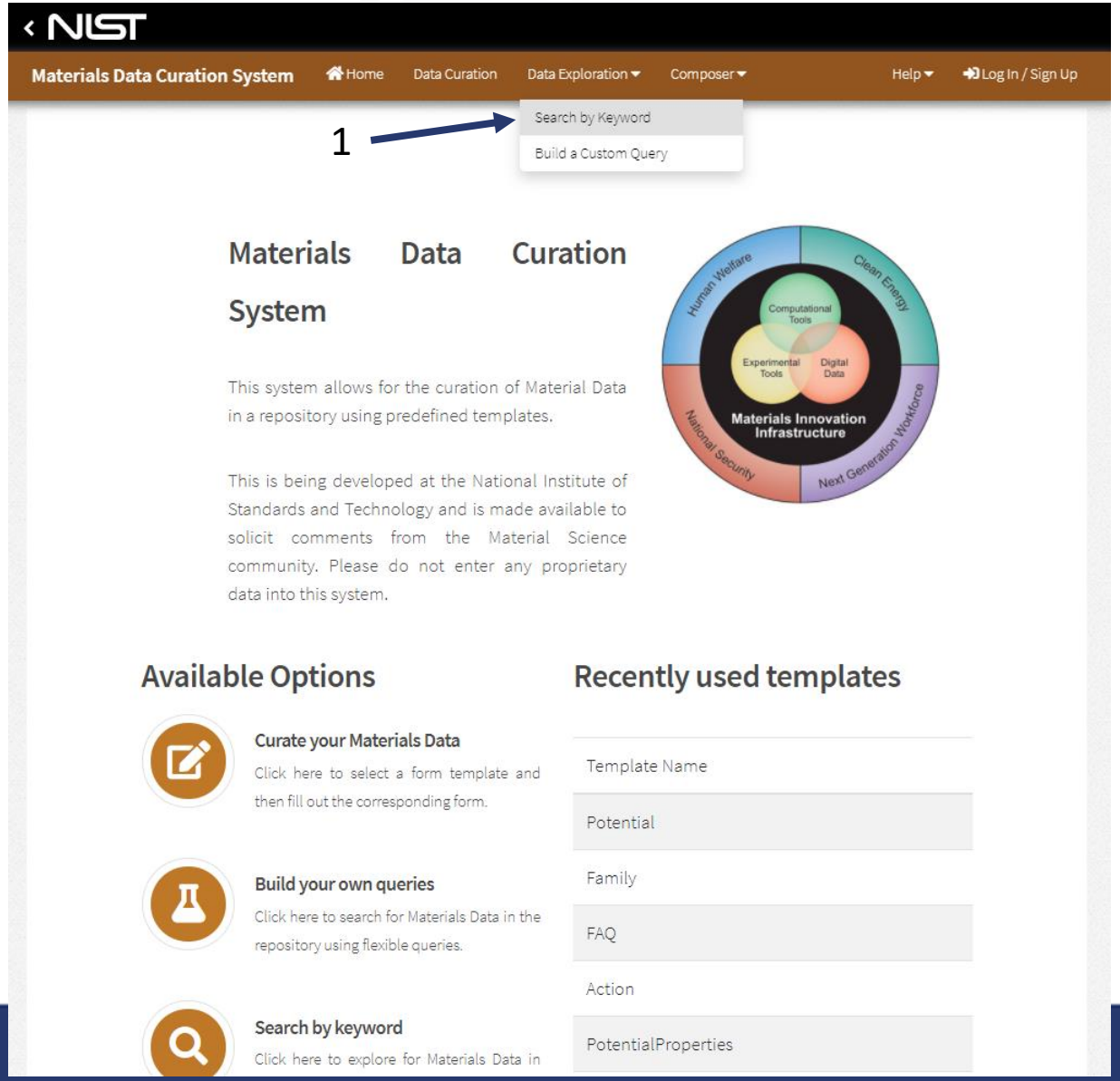
CDCS DATABASE: HTTPS://POTENTIALS.NIST.GOV

Records used to generate Repository
Publicly accessible web-based search
REST API (email potentials@nist.gov)

Quick search guide:

1. Click Data Exploration->Search by keyword
2. Check Potential in Filter by Template
3. Enter any keywords in the box
4. Click Search
5. Click + to expand, or click name to open

Working on making it prettier
Calculation results coming soon



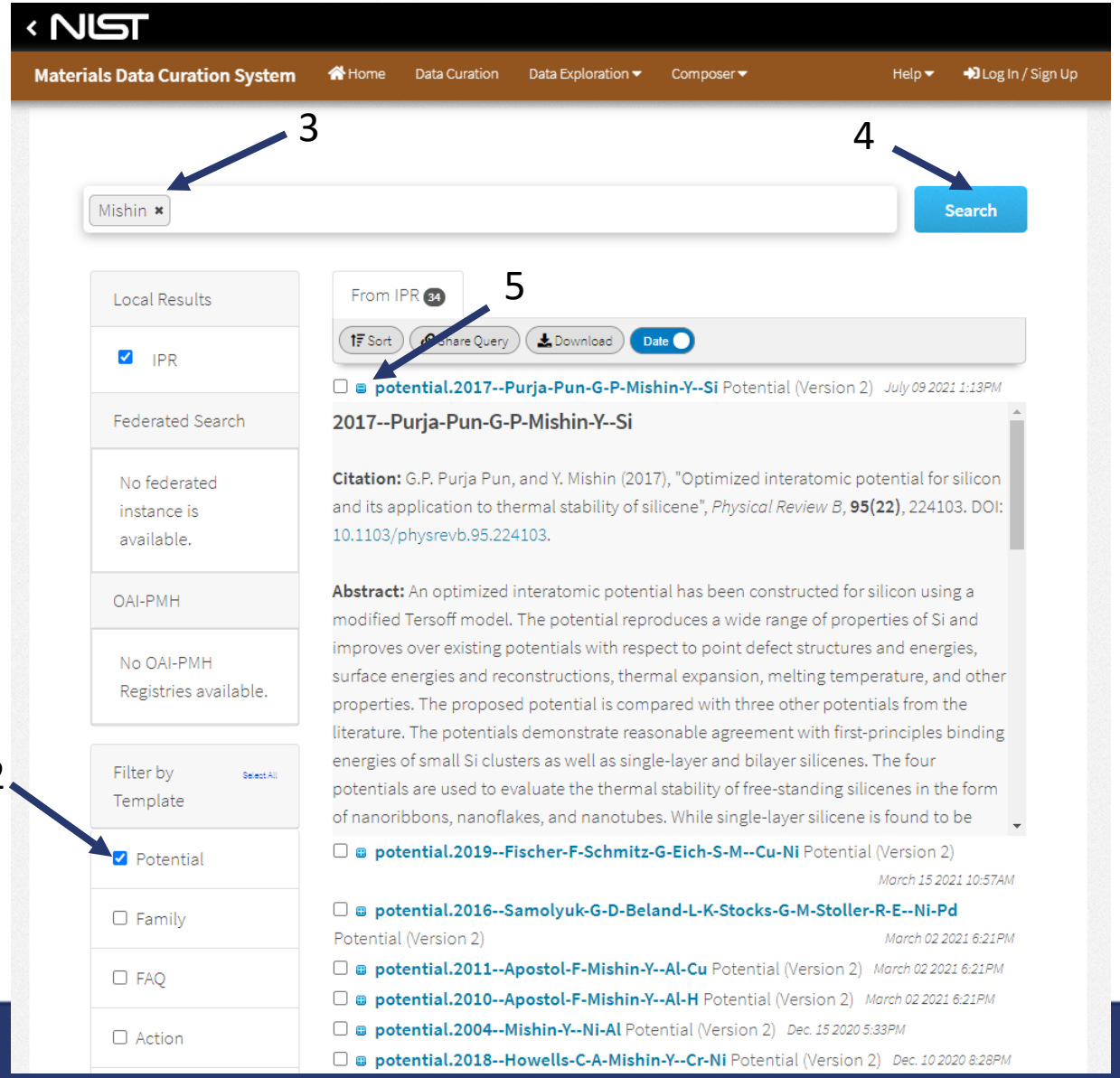
CDCS DATABASE: HTTPS://POTENTIALS.NIST.GOV

Records used to generate Repository
Publicly accessible web-based search
REST API (email potentials@nist.gov)

Quick search guide:

1. Click Data Exploration->Search by keyword
2. Check Potential in Filter by Template
3. Enter any keywords in the box
4. Click Search
5. Click + to expand, or click name to open

Working on making it prettier
Calculation results coming soon

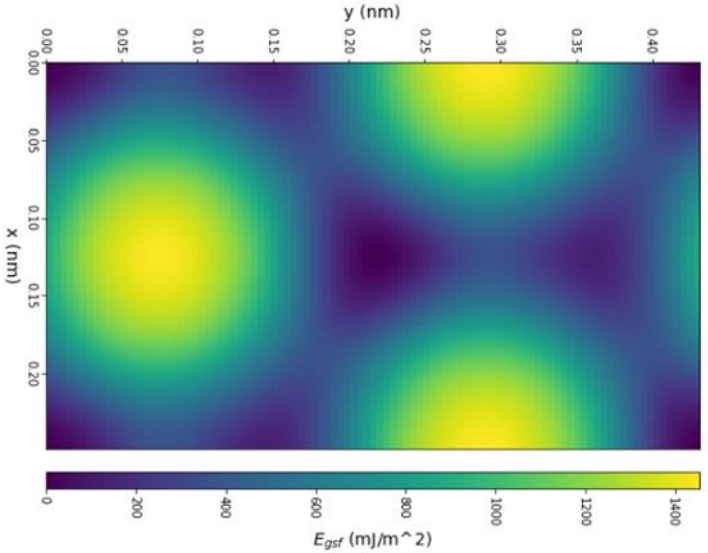


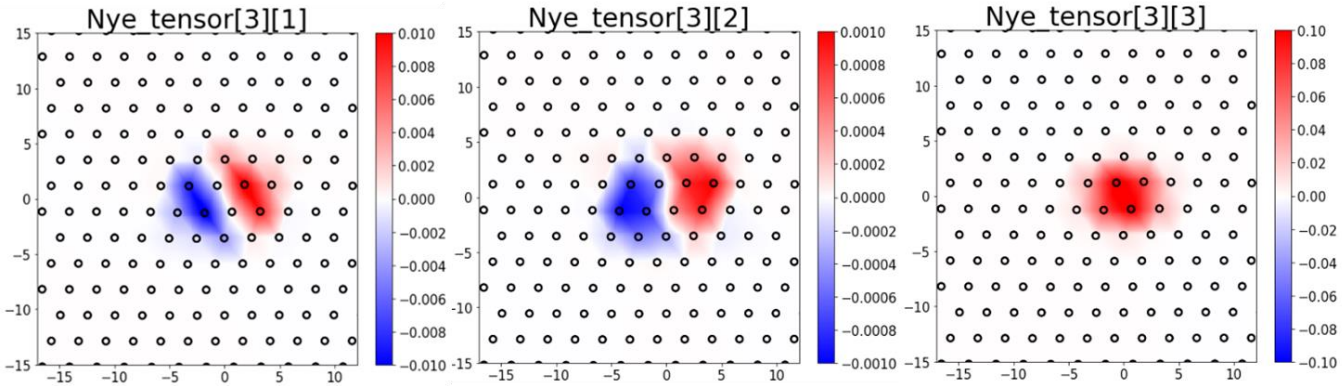
ATOMMAN : ATOMISTIC MANIPULATION TOOLKIT

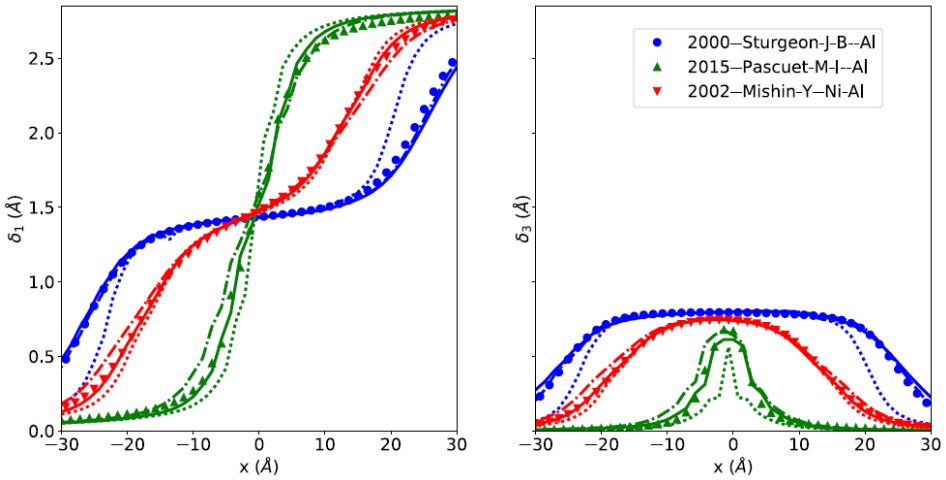
https://github.com/usnistgov/atomman

https://www.ctcms.nist.gov/potentials/atomman

- Search and download any LAMMPS potentials from the repository
- Load, build and manipulate atomic systems
- Create crystal defect configurations using built in tools
- Run LAMMPS from within Python and parse results
- Analyze relaxed crystal defects







SEARCH FOR POTENTIALS IN PYTHON

Explore available potentials

```
# Initialize link to database
potdb = am.library.Database()

# Get locally-hosted Potentials listings
potentials, potentials_df = potdb.get_potentials(remote=False, return_df=True, verbose=True)
```

```
Found 649 matching Potential records in local library
```

```
potdb.widget_search_potentials(potentials, potentials_df)
```

Search potential listings

Element1: Al
Element2:
Element3: Cu
Year:
Author:
Potential: 1996--Cai-J-Ye-Y-Y--Al-Cu

1996--Cai-J-Ye-Y-Y--Al-Cu
1996--Jacobsen-K-W-Stoltze-P-Norskov-J-K--Al-Ag-Au-Cu-Ni-Pd-Pt
1999--Liu-X-Y-Liu-C-L-Borucki-L-J--Al-Cu
2011--Apostol-F-Mishin-Y--Al-Cu
2012--Jelinek-B-Groh-S-Horstemeyer-M-F-et-al--Al-Si-Mg-Cu-Fe
2016--Zhou-X-W-Ward-D-K-Foster-M-E--Al-Cu
2018--Zhou-X-W-Ward-D-K-Foster-M-E--Al-Cu-H

1996--Cai

Citation: J. Ca… …odel including a long-range force for fcc metals and the… …evb.54.8398.

Abstract: A si… …. The model includes a long-range force. In this model, the electron-density function is taken as a decreasing exponential function, the two-body potential is defined as a function like a form given by Rose *et al.* [Phys. Rev. B **33**, 7983 (1986)], and the embedding energy is assumed to be an universal form recently suggested by Banerjea and Smith. The embedding energy has a positive curvature. The model is applied to seven fcc metals (Al, Ag, Au, Cu, Ni, Pd, and Pt) and their binary alloys. All the considered properties, whether for pure metal systems or for alloy systems, are predicted to be satisfactory at least qualitatively. The model resolves the problems of Johnson's model for predicting the properties of the alloys involving metal Pd. However, more importantly, (i) by investigating the structure stability of seven fcc metals using the present model, we found that the stability energy is dominated by both the embedding energy and the pair potential for fcc-bcc stability while the pair potential dominates and is underestimated for fcc-hcp stability; and (ii) we find that the predicted total energy as a function of lattice parameter is in good agreement with the equation of state of Rose et al. for all seven fcc metals, and that this agreement is closely related to the electron density, i.e., the lower the contribution from atoms of the second-nearest neighbor to host density, the better the agreement becomes. We conclude the following: (i) for an EAM, where angle force is not considered, the long-range force is necessary for a prediction of the structure stability; or (ii) the dependence of the electron density on angle should be considered so as to improve the structure-stability energy. The conclusions are valid for all EAM models where an angle force is not considered.

OpenKIM (MO_942551040047)

Fetch LAMMPS potential

```
# Get locally-hosted LAMMPS Potentials listings
lammps_potentials, lammps_potentials_df = potdb.get_lammps_potentials(remote=False, return_df=True,
                                                                      pot_dir_style='local', verbose=True)
```

```
Found 408 matching potential_LAMMPS records in local library
Found 450 matching potential_LAMMPS_KIM records in local library
Built 302 lammps potentials for KIM models
```

```
results = {}
potdb.widget_lammps_potential(lammps_potentials, lammps_potentials_df, results)
```

Select a LAMMPS potential

Use the dropdown boxes to parse and select a potential. If you wish to copy/download the parameter files to the current working directory then click "Copy Files" after selection.

Element1: Al
Element2: Cu
Element3:
Pair Style:
Potential: 2016--Zhou-X-W--Al-Cu--LAMM

Copy Files

```
potential = results['lammps_potential']
print(potential.id)
```

```
2016--Zhou-X-W--Al-Cu--LAMMPS--ipr2
```

```
print(potential.pair_info(['Cu', 'Al', 'Cu']))
```

```
print "Potential 2016--Zhou-X-W--Al-Cu--LAMMPS--ipr2 listed in the NIST Interatomic Potentials Repository:"
print "https://www.ctcms.nist.gov/potentials/entry/2016--Zhou-X-W-Ward-D-K-Foster-M-E--Al-Cu/2016--Zhou-X-W--Al-Cu--LAMMPS--ipr2.html"
print "Publication(s) related to the potential:"
print "https://doi.org/10.1016/j.jallcom.2016.04.055"
print "Parameter file(s) can be downloaded at:"
print "https://www.ctcms.nist.gov/potentials/Download/2016--Zhou-X-W-Ward-D-K-Foster-M-E--Al-Cu/2/AlCu.bop.table"
pair_style bop
pair_coeff * * 2016--Zhou-X-W--Al-Cu--LAMMPS--ipr2\AlCu.bop.table Cu Al Cu
mass 1 63.55
mass 2 26.98
mass 3 63.55

comm_modify cutoff 14.7
```

Or, without GUI

```
import atomman as am

potential = am.load_lammps_potential(id='2004--Zhou-X-W--Au--LAMMPS--ipr2')
print(potential.id)
```

```
2004--Zhou-X-W--Au--LAMMPS--ipr2
```

LOAD AND MANIPULATE SYSTEM

Fetch a relaxed unit cell from the database

```
ucell = am.load('crystal', potential=potential)
print()
print(ucell)
```

```
Multiple matching record retrieved from remote
#  family                symbols  alat    Ecoh    method  standing
 1 A1--Cu--fcc           Au        4.0801 -3.9300 dynamic good
 2 A3'--alpha-La--doubl Au        2.8800 -3.9296 dynamic good
 3 oqmd-1216020          Au        2.8787 -3.9296 dynamic good
 4 A3--Mg--hcp           Au        2.8760 -3.9295 dynamic good
 5 oqmd-1214862          Au        7.0330 -3.8761 dynamic good
 6 oqmd-1214773          Au       10.0234 -3.8759 dynamic good
 7 oqmd-1038224          Au        4.6584 -3.8702 dynamic good
 8 A15--beta-W           Au        5.1768 -3.8613 dynamic good
Please select one: 1

avect =  [ 4.080,  0.000,  0.000]
bvect =  [ 0.000,  4.080,  0.000]
cvect =  [ 0.000,  0.000,  4.080]
origin = [ 0.000,  0.000,  0.000]
natoms = 4
natypes = 1
symbols = ('Au',)
pbc = [ True  True  True]
per-atom properties = ['atype', 'pos']
     id |   atype |  pos[0] |  pos[1] |  pos[2]
      0 |       1 |   0.000 |   0.000 |   0.000
      1 |       1 |   0.000 |   2.040 |   2.040
      2 |       1 |   2.040 |   0.000 |   2.040
      3 |       1 |   2.040 |   2.040 |   0.000
```

```
rcell = ucell.rotate([[1,1,1], [1,-1, 0], [1,1,-2]])
print(rcell)
```

```
avect =  [ 7.067,  0.000,  0.000]
bvect =  [ 0.000,  5.770,  0.000]
cvect =  [ 0.000,  0.000,  9.994]
origin = [ 0.000,  0.000,  0.000]
natoms = 24
natypes = 1
symbols = ('Au',)
pbc = [ True  True  True]
per-atom properties = ['atype', 'pos']
     id |   atype |  pos[0] |  pos[1] |  pos[2]
      0 |       1 |   7.067 |   1.443 |   7.496
      1 |       1 |   7.067 |   4.328 |   7.496
      2 |       1 |   2.356 |   4.328 |   9.161
      3 |       1 |   2.356 |   1.443 |   9.161
      4 |       1 |   7.067 |   4.328 |   2.499
      5 |       1 |   7.067 |   1.443 |   2.499
      6 |       1 |   7.067 |   5.770 |   4.997
      7 |       1 |   0.000 |   2.885 |   4.997
      8 |       1 |   2.356 |   1.443 |   4.164
      9 |       1 |   2.356 |   4.328 |   4.164
     10 |       1 |   2.356 |   2.885 |   6.663
     11 |       1 |   4.711 |   4.328 |   5.830
     12 |       1 |   2.356 |   5.770 |   6.663
     13 |       1 |   4.711 |   1.443 |   5.830
     14 |       1 |   4.711 |   5.770 |   8.328
     15 |       1 |   4.711 |   2.885 |   8.328
     16 |       1 |   7.067 |   2.885 |   9.994
     17 |       1 |   0.000 |   0.000 |   9.994
     18 |       1 |   2.356 |   5.770 |   1.666
     19 |       1 |   2.356 |   2.885 |   1.666
     20 |       1 |   4.711 |   1.443 |   0.833
     21 |       1 |   4.711 |   4.328 |   0.833
     22 |       1 |   4.711 |   2.885 |   3.331
     23 |       1 |   4.711 |   5.770 |   3.331
```

Rotate ensures periodic dimensions remain compatible

```
system = rcell.supersize(100, (-20, 20), (-10, 10))
print(system.box)
print(system.natoms)
```

```
avect =  [706.686,  0.000,  0.000]
bvect =  [ 0.000, 230.803,  0.000]
cvect =  [ 0.000,  0.000, 199.881]
origin = [ 0.000, -115.401, -99.940]
1920000
```

Replicate in both +- directions

EXAMPLE: NANOWIRE

Create nanowire or nanopillar

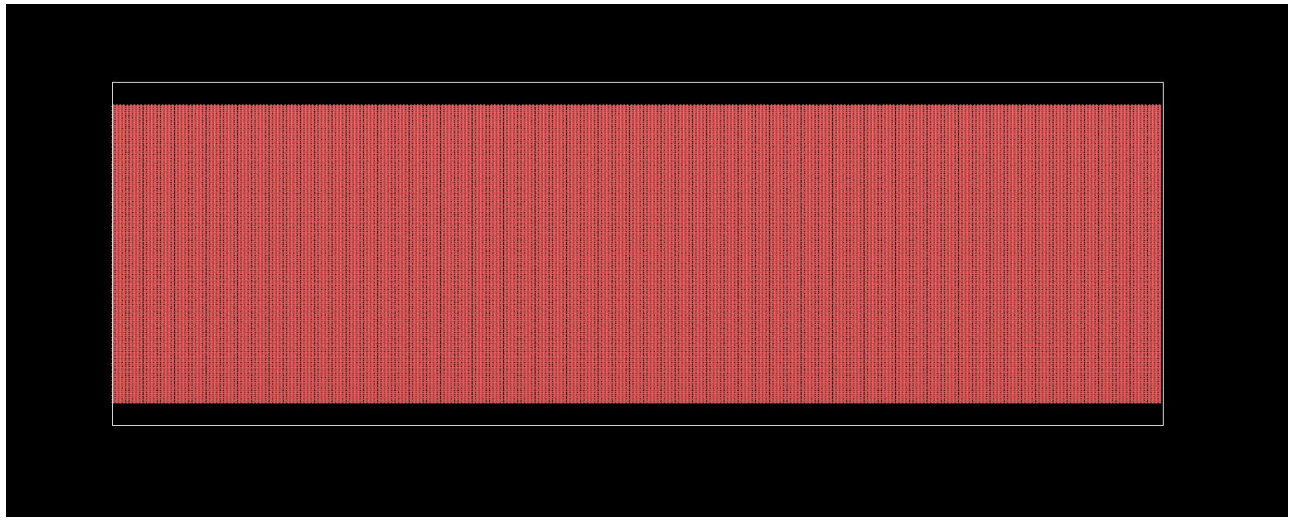
```
center1 = [0,0,0]
center2 = [system.box.a, 0,0]
radius = 100

cylinder = am.region.Cylinder(center1, center2, radius)
```

```
cylinder.inside(system.atoms.pos).sum()
```

```
1309000
```

```
pillar = system.atoms_ix[cylinder.inside(system.atoms.pos)]
```

```
pillar.pbc = (False, False, False)
```

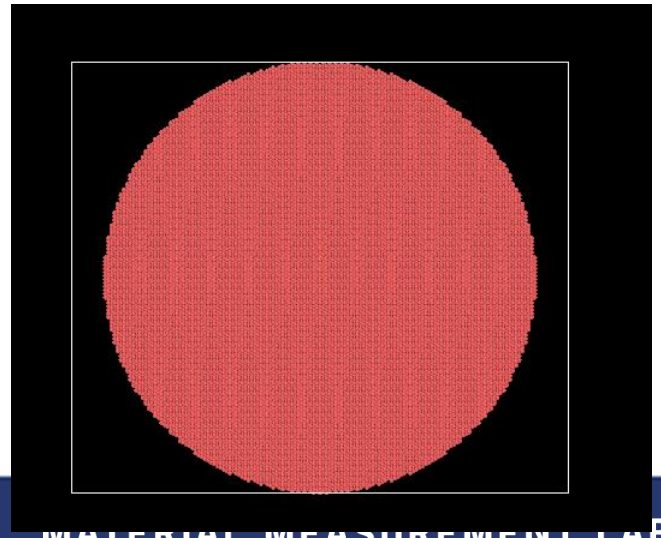


Save

```
system_pair_info = pillar.dump('atom_data', f='initial.dat',
                               potential=potential,
                               return_pair_info=True)
```

```
print(system_pair_info)
```

```
# Script and atom data file prepared using atomman Python package

units metal
atom_style atomic

boundary m m m
read_data initial.dat

mass 1 196.97

pair_style eam/alloy
pair_coeff * * 2004--Zhou-X-W--Au--LAMMPS--ipr2\Au_Zhou04.eam.alloy Au
```



SETUP AND RUN LAMMPS

Template LAMMPS Script

```
script_template = """
<system_pair_info>

variable temp equal <temperature>

velocity all create ${temp} <randomseed>

compute pe all pe/atom
compute ke all ke/atom
compute stress all stress/atom NULL

thermo <thermosteps>
thermo_style custom step pe temp

dump dumpit all custom <dumpsteps> *.dump id type xu yu zu c_pe c_ke &
    c_stress[1] c_stress[2] c_stress[3] c_stress[4] c_stress[5] c_stress[6]
dump_modify dumpit format float %.13e

restart <restartsteps> *.restart

timestep 0.01

fix 1 all nvt temp ${temp} ${temp} 1.0
run <runsteps>
"""
```

Calculation settings

```
lammps_parameters = {}
lammps_parameters['system_pair_info'] = system_pair_info
lammps_parameters['temperature'] = '100'
lammps_parameters['randomseed'] = str(np.random.randint(1, 900000000))
lammps_parameters['thermosteps'] = '100'
lammps_parameters['restartsteps'] = '10000'
lammps_parameters['dumpsteps'] = '10000'
lammps_parameters['runsteps'] = '100000'

script = am.tools.filltemplate(script_template, lammps_parameters, '<', '>')

with open('start.in', 'w') as f:
    f.write(script)

print(script)
```

Full LAMMPS Script

```
# Script and atom data file prepared using atomman Python package

units metal
atom_style atomic

boundary m m m
read_data initial.dat

mass 1 196.97

pair_style eam/alloy
pair_coeff * * 2004--Zhou-X-W--Au--LAMMPS--ipr2\Au_Zhou04.eam.alloy Au


variable temp equal 100

velocity all create ${temp} 893638779

compute pe all pe/atom
compute ke all ke/atom
compute stress all stress/atom NULL

thermo 100
thermo_style custom step pe temp

dump dumpit all custom 10000 *.dump id type xu yu zu c_pe c_ke &
    c_stress[1] c_stress[2] c_stress[3] c_stress[4] c_stress[5] c_stress[6]
dump_modify dumpit format float %.13e

restart 10000 *.restart

timestep 0.01

fix 1 all nvt temp ${temp} ${temp} 1.0
run 100000
```

Run and view results

```
log = am.lammps.run('lmp_mpi', 'start.in', mpi_command='mpiexec -localonly 6')
```

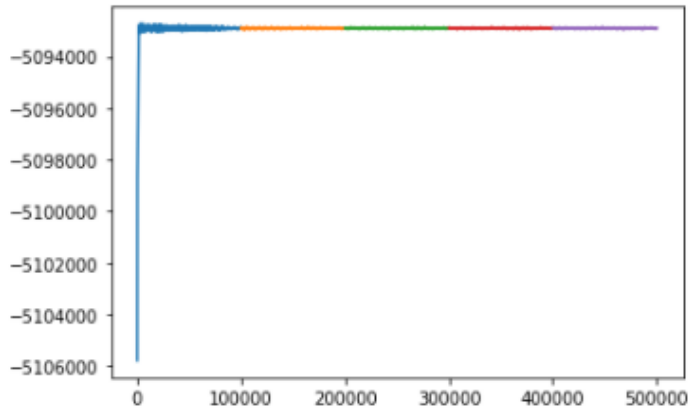
```
log.simulations[0].performance
```

Section	min time	avg time	max time	%varavg	%total
Pair	32264.000	32561.000	32753.000	104.7	94.03
Neigh	27.753	28.115	28.451	5.0	0.08
Comm	430.430	582.560	868.240	580.3	1.68
Output	24.098	24.110	24.128	0.2	0.07
Modify	1046.000	1139.600	1326.300	282.6	3.29
Other	0.000	291.600	0.000	0.0	0.84

```
log.simulations[0].thermo
```

	Step	PotEng	Temp
0	0	-5105778.1	100.000000
1	100	-5098961.4	62.763689
2	200	-5098711.8	65.291027
3	300	-5098340.3	67.595614
4	400	-5097945.5	70.120858
5	500	-5097526.6	72.788167
6	600	-5097058.5	75.378864
7	700	-5096699.4	78.754788
8	800	-5096176.4	81.246197

```
for sim in log.simulations:
    plt.plot(sim.thermo.Step, sim.thermo.PotEng)
```



IPRPY

https://github.com/usnistgov/iprPy

https://www.ctcms.nist.gov/potentials/iprPy/

Collects the property calculations used by the NIST Interatomic Potentials Repository

Focus on making the calculations as accessible as possible at all levels

High-throughput workflow tools to run calculations *en masse*

NEW VERSION APPROACHES!

- https://github.com/lmhale99/iprPy
- pip/conda installable
- Options to run calculations entirely from .txt and command lines
- Results to be added to the CDCS site – iprPy will allow searches
- Better modularity for calculations = easier to add new ones
- Docs are out of date: email potentials@nist.gov for help

Load potential and point to local database for parameter file(s)

```
potential = am.load_lammps_potential(id='2016--Stoller-R-E--Ni--LAMMPS--ipr1',
                                     pot_dir_style='local')
```

Load unit cell prototype and create a larger test system

```
ucell = am.load_prototype('A1--Cu--fcc', symbols='Ni')
system = ucell.supersize(3,3,3)
```

Load the E_vs_r_scan calculation

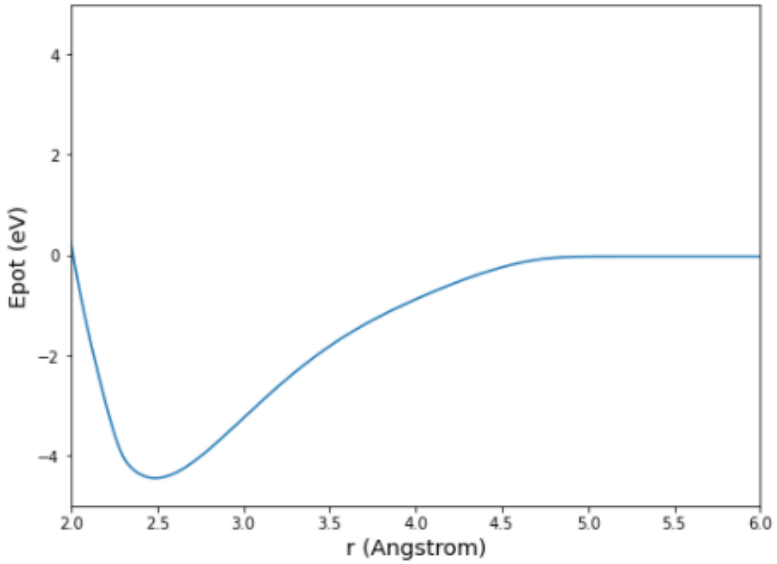
```
calc = iprPy.load_calculation('E_vs_r_scan')
```

Run calculation

```
results = calc.calc('lmp_mpi', system, potential, ucell=ucell,
                    rmin=2.0, rmax=6.0,  rsteps=200)
```

Plot results

```
fig = plt.figure(figsize=(8,6))
plt.plot(results['r_values'], results['Ecoh_values'])
plt.xlim(2, 6)
plt.ylim(-5, 5)
plt.xlabel('r (Angstrom)', size='x-large')
plt.ylabel('Epot (eV)', size='x-large')
plt.show()
```



RUN USING COMMAND LINE AND TEXT FILES

1. Copy empty input script

```
$ iprPy template isolated_atom
calc_isolated_atom.in created
```

2. View input script's terms

```
$ iprPy templatedoc isolated_atom
# isolated_atom Input Terms

## Calculation Metadata

Specifies metadata descriptors common to all calculation st

- __branch__: A metadata group name that the calculation ca
sed by. Primarily meant for differentiating runs with diffe
tings parameters.

## LAMMPS and MPI Commands

Specifies the external commands for running LAMMPS and MPI.

- __lammps_command__: The path to the executable for runnir
 on your system. Don't include command line options.
- __mpi_command__: The path to the MPI executable and any c
ine options to use for calling LAMMPS to run in parallel or
stem. LAMMPS will run as a serial process if not given.

## Interatomic Potential

Specifies the interatomic potential to use and the director
any associated parameter files are located.

- __potential_file__: The path to the potential_LAMMPS or p
_LAMMPS_KIM record that defines the interatomic potential t
r LAMMPS calculations.
- __potential_kim_id__: If potential_file is a potential_LA
 record, this allows for the specification of which versior
KIM model to use by specifying a full kim model id.  If not
the newest known version of the kim model will be assumed.
```

3. Copy/download LAMMPS potential

```
$ iprPy get_lammps_potential 2003--Lee-B-J--Ag--LAMMPS--ipr1 -c
Found 1 matching potential_LAMMPS records in local library
Found 0 matching potential_LAMMPS_KIM records in local library
Built 0 lammps potentials for KIM models
Matching record retrieved from local

2003--Lee-B-J--Ag--LAMMPS--ipr1.json copied to working directory
library.meam copied to 2003--Lee-B-J--Ag--LAMMPS--ipr1
Ag.meam copied to 2003--Lee-B-J--Ag--LAMMPS--ipr1
potential_dir is: 2003--Lee-B-J--Ag--LAMMPS--ipr1
```

4. Fill in input script

```
# Input script for iprPy calculation isolated_atom

# Calculation Metadata
branch

# LAMMPS and MPI Commands
lammps_command                  lmp_mpi
mpi_command

# Interatomic Potential
potential_file                  2003--Lee-B-J--Ag--LAMMPS--ipr1.json
potential_kim_id
potential_dir                   2003--Lee-B-J--Ag--LAMMPS--ipr1

# Input/Output Units
length_unit
pressure_unit
energy_unit
force_unit
```

5. Run calculation from the script

```
$ iprPy run calc_isolated_atom.in
Calculation finished successfully
```

6. View results.json


```
"calculation-isolated-atom": {
    "key": "2d6e32f0-a1b9-4857-a4e4-d593a979713d",
    "calculation": {
        "iprPy-version": "0.11.0",
        "atomman-version": "1.4.0",
        "LAMMPS-version": "3 Mar 2020",
        "script": "calc_isolated_atom",
        "branch": "main"
    },
    "potential-LAMMPS": {
        "key": "d1069118-cfb7-4499-8557-0204b7054093",
        "id": "2003--Lee-B-J--Ag--LAMMPS--ipr1",
        "potential": {
            "key": "08baaee0-0c9b-4219-8b17-2b009ff00867",
            "id": "2003--Lee-B-J-Shim-J-H-Baskes-M-I--Ag"
        }
    },
    "isolated-atom-energy": {
        "symbol": "Ag",
        "energy": {
            "value": 0.0,
            "unit": "eV"
        }
    }
}
```

HIGH-THROUGHPUT CALCULATIONS

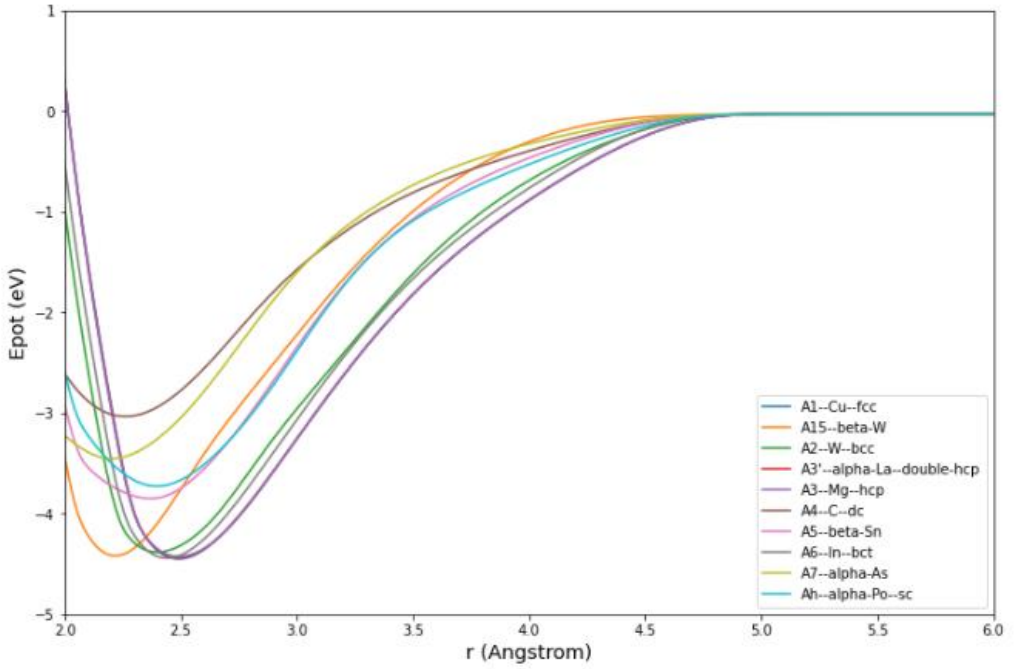
```
# Input script for master_prepare.py

# Name of the database to prepare the calculations for
database_name               iprhub

# Machine-dependent mpi command with variable np_per_runner
mpi_command                 /cluster/deb9/bin/mpirun -n {np_per_runner}

# Primary lammps executable
lammps_command              lmp_mpi

# Optional alternate lammps executables
lammps_command_snap_1       /users/lmh1/LAMMPS/bin/lmp_mpi_2017_03_31
lammps_command_snap_2       /users/lmh1/LAMMPS/bin/lmp_mpi_2019_06_05
lammps_command_old          /users/lmh1/LAMMPS/bin/lmp_mpi_2019_06_05
lammps_command_aenet        /users/lmh1/LAMMPS/bin/lmp_mpi_2020_03_03_aenet
lammps_command_kim          /users/lmh1/LAMMPS/bin/lmp_mpi_2020_03_03_kim

# Indicate if lammps executables are to be tested (default is True)
test_commands               True

# List of calculation styles to run_directory pools
# Must have equal number of lines for run_directory_name, np_per_runner and styles

styles                      isolated_atom diatom_scan E_vs_r_scan
run_directory_name          iprhub_3
np_per_runner               1

styles                      relax_box relax_dynamic relax_static
run_directory_name          iprhub_4
np_per_runner               1

styles                      relax_static_from_dynamic
run_directory_name          iprhub_5
np_per_runner               1

styles                      crystal_space_group
run_directory_name          iprhub_6
np_per_runner               1
```

```
iprhub = iprPy.load_database('iprhub')
```

```
records_df = iprhub.get_records_df('calculation_E_vs_r_scan', potential_LAMMPS_id=potential.id)
len(records_df)
```

```
10
```

```
fig = plt.figure(figsize=(12,8))
for i in records_df.sort_values('family').index:
    series = records_df.loc[i]
    plt.plot(series.r_values, series.energy_values, label=series.family)
plt.legend()
plt.xlim(2, 6)
plt.ylim(-5, 1)
plt.xlabel('r (Angstrom)', size='x-large')
plt.ylabel('Epot (eV)', size='x-large')
plt.show()
```



THANKS!

Find, download and explore potentials content:

- Main repository website: https://www.ctcms.nist.gov/potentials/
- CDCS database: https://potentials.nist.gov
- github JSON record dump: https://github.com/lmhale99/potentials-library

atomman links

- documentation: https://www.ctcms.nist.gov/potentials/atomman
- code: https://github.com/usnistgov/atomman

iprPy

- New code: https://github.com/lmhale99/iprPy (email for help)

Submit potentials, ask questions, get help

- Email potentials@nist.gov