



Interatomic Potentials Repository Project

Lucas M. Hale
Zachary T. Trautt
Chandler A. Becker

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

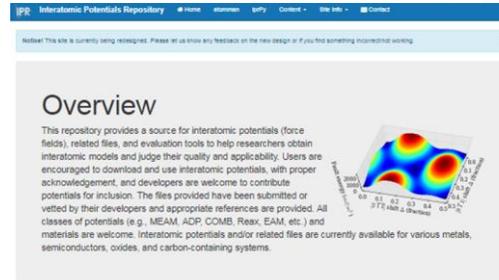
IPR

**MATERIAL
MEASUREMENT
LABORATORY**

MAJOR COMPONENTS OF IPR

NIST Interatomic Potentials Repository [1]

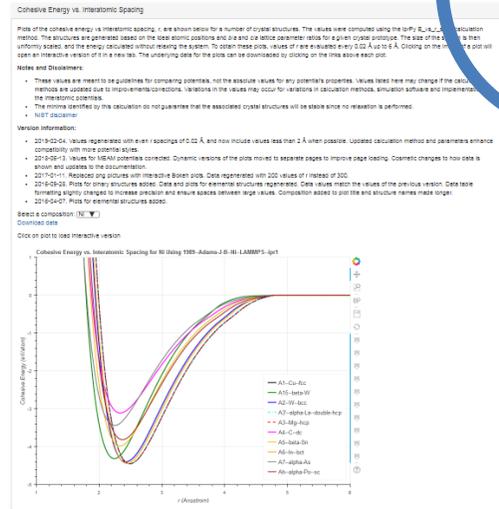
- Hosts +290 different interatomic potentials
- Computed properties for comparison



Interatomic Potentials (Force fields)

Elements

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



```
def relax_dynamic(lammps_command, system, potential, mpi_command=None,
                 p_xx=0.0, p_yy=0.0, p_zz=0.0, p_xy=0.0, p_xz=0.0, p_yz=0.0,
                 temperature=0.0, integrator=None, runsteps=220000,
                 thermostat=100, dumpsteps=None, equilsteps=20000,
                 randomseed=None):
    try:
        # Get script's location if __file__ exists
        script_dir = Path(__file__).parent
    except:
        # Use cwd otherwise
        script_dir = Path.cwd()

    # Get lammps units
    lammps_units = Imp.style.unit(potential.units)

    # Get lammps version date
    lammps_date = Imp.checkversion(lammps_command) ['date']

    # Handle default values
    if dumpsteps is None:
        dumpsteps = runsteps

    # Define lammps variables
    lammps_variables = {}
    system_info = system.dump('atom_data', f='init.dat',
                             units=potential.units,
                             atom_style=potential.atom_style)
    lammps_variables['atomman_system_info'] = system_info
    lammps_variables['atomman_pair_info'] = potential.pair_info(system.symbols)
    integ_info = integrator_info(integrator=integrator,
                                p_xx=p_xx, p_yy=p_yy, p_zz=p_zz,
                                p_xy=p_xy, p_xz=p_xz, p_yz=p_yz,
                                temperature=temperature,
                                randomseed=randomseed,
                                units=potential.units)
```

atomman Python package [2]

- Create and manipulate atomic configurations
- Interact with LAMMPS MD software [3]
- Focus on crystalline defects

iprPy Python package [2]

- Complete property calculations
- High-throughput workflow tools

0000d14-9c42-48f9-8617-3fd2fe69f23f	8/26/2019 3:16 PM	File folder
001c360-242f-4d96-bb6d-41876cb9f823	8/26/2019 3:16 PM	File folder
00b878b7-d1b4-4379-9400-4616028ebdf5	8/26/2019 3:18 PM	File folder
00c02348-2d9e-4c48-9f1f-967e54931073	8/26/2019 3:16 PM	File folder
00d4b222-9b51-4213-9892-881566fb1308	8/26/2019 3:16 PM	File folder
00e7d306-fad4-482a-9b6c-fd5af5536456	8/26/2019 3:18 PM	File folder
00e512b6-fa8d-451b-9f74-4fd3eeb60232	8/26/2019 3:16 PM	File folder
00ee877a-7357-4a1c-a1bf-a5249b3d5b13	8/26/2019 3:19 PM	File folder
00efe511-c27d-4d5e-a9b0-6ec5aee58a52	8/26/2019 3:15 PM	File folder
00fd534a-c3b8-4b67-ae5f-7e0e0df7f4be	8/26/2019 3:17 PM	File folder
00ff75f0-eb7e-4505-a0c3-4c02ff3834bd	8/26/2019 3:16 PM	File folder
0a0ed84c-3c6b-4c56-8462-d81cb8ffd3ed	8/26/2019 3:18 PM	File folder
0a4fa1d9-7702-4876-9508-64821b9dea59	8/26/2019 3:16 PM	File folder
0a5d330-f298-494c-b8b1-3e804ad1606f	8/26/2019 3:16 PM	File folder
0a7fb0c7-bda3-425a-8d2b-90d797cbe46b	8/26/2019 3:16 PM	File folder
0a13f512-b03e-475b-96c0-40fd1adb8add	8/26/2019 3:18 PM	File folder
0a48bbf9-4505-43ba-a244-1b58655014cd	8/26/2019 3:18 PM	File folder
0a77e10a-d1f2-480b-bd95-47226ff19f9a	8/26/2019 3:16 PM	File folder

[1] C.A. Becker, F. Tavazza, Z.T. Trautt, and R.A. Buarque de Macedoc (2013) *Curr Opin Solid State Mater Sci*, **17**, 277-283.
 [2] L.M. Hale, Z.T. Trautt, and C.A. Becker (2018) *Model Simul Mat Sci Eng*, **26**, 055003.
 [3] S. Plimpton (1995) *J Comp Phys*, **117**, 1-19

WHY ARCHIVE INTERATOMIC POTENTIAL FILES?

Easier to discover existing models - developing/training is time consuming

Newest is not “best” - different potentials trained for different conditions

Publications alone are often not enough to reproduce models

- Typos, errors and missing parameters
- Parameter rounding, numerical tabulations

Hosted models and their publications gain more exposure

(Submit models by emailing potentials@nist.gov)

INTERATOMIC POTENTIALS REPOSITORY

<https://www.ctcms.nist.gov/potentials/>

290+ potentials

- Known provenance
- Any format
 - EAM, MEAM, 3-Body, ML, reactive
 - FORTRAN, pdfs, LAMMPS mods
- Full citation and abstracts
- Can list potentials hosted elsewhere
- Property calculations

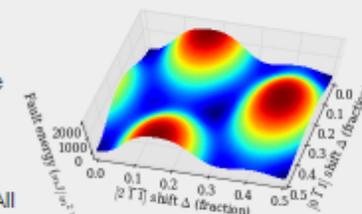
Incorporated into many projects

- OpenKIM
- JARVIS-FF
- Pyiron
- atomistictools.org
- MedeA

Notice! This site is currently being redesigned. Please let us know any feedback on the new design or if you find something incorrect/not working.

Overview

This repository provides a source for interatomic potentials (force fields), related files, and evaluation tools to help researchers obtain interatomic models and judge their quality and applicability. Users are encouraged to download and use interatomic potentials, with proper acknowledgement, and developers are welcome to contribute potentials for inclusion. The files provided have been submitted or vetted by their developers and appropriate references are provided. All classes of potentials (e.g., MEAM, ADP, COMB, Reax, EAM, etc.) and materials are welcome. Interatomic potentials and/or related files are currently available for various metals, semiconductors, oxides, and carbon-containing systems.



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

INTERATOMIC POTENTIALS REPOSITORY

<https://www.ctcms.nist.gov/potentials/>

290+ potentials

- Known provenance
- Any format
 - EAM, MEAM, 3-Body, ML, reactive
 - FORTRAN, pdfs, LAMMPS mods
- Full citation and abstracts
- Can list potentials hosted elsewhere
- Property calculations

Incorporated into many projects

- OpenKIM
- JARVIS-FF
- Pyiron
- atomistictools.org
- MedeA

2006--Williams-P-L-Mishin-Y-Hamilton-J-C--Ag

Citation: P.L. Williams, Y. Mishin, and J.C. Hamilton (2006), "An embedded-atom potential for the Cu-Ag system", *Modelling and Simulation in Materials Science and Engineering*, 14(5), 817-833. DOI: [10.1088/0965-0393/14/5/002](https://doi.org/10.1088/0965-0393/14/5/002).

Abstract: A new embedded-atom method (EAM) potential has been constructed for Ag by fitting to experimental and first-principles data. The potential accurately reproduces the lattice parameter, cohesive energy, elastic constants, phonon frequencies, thermal expansion, lattice-defect energies, as well as energies of alternate structures of Ag. Combining this potential with an existing EAM potential for Cu, a binary potential set for the Cu-Ag system has been constructed by fitting the cross-interaction function to first-principles energies of imaginary Cu-Ag compounds. Although properties used in the fit refer to the 0 K temperature (except for thermal expansion factors of pure Cu and Ag) and do not include liquid configurations, the potentials demonstrate good transferability to high-temperature properties. In particular, the entire Cu-Ag phase diagram calculated with the new potentials in conjunction with Monte Carlo simulations is in satisfactory agreement with experiment. This agreement suggests that EAM potentials accurately fit to 0 K properties can be capable of correctly predicting simple phase diagrams. Possible applications of the new potential set are outlined.

EAM tabulated functions

Notes: These files were provided by Yuri Mishin.

File(s):

Ag F(ρ): [F_ag.plt](#)

Ag ρ (r): [fag.plt](#)

Ag ϕ (r): [pag.plt](#)

LAMMPS pair_style eam/alloy (2006--Williams-P-L--Ag--LAMMPS--ipr1)

[See Computed Properties](#)

Notes: This conversion was produced by Chandler Becker on 4 February 2009 from the plt files listed above. This version is compatible with LAMMPS. Validation and usage information can be found in [Ag06_releaseNotes_1.pdf](#). If you use this setfile, please credit the website in addition to the original reference.

File(s):

[Ag.eam.alloy](#)

[Ag06_releaseNotes_1.pdf](#)

PROPERTY CALCULATIONS

Diatom Energy vs. Interatomic Spacing

Plots of the potential energy vs interatomic spacing, r , are shown below for all diatom sets associated with the interatomic potential. This calculation provides insights into the functional form of the potential's two-body interactions. A system consisting of only two atoms is created, and the potential energy is evaluated for the atoms separated by $0.02 \text{ \AA} \leq r \leq 6.0$ in intervals of 0.02 \AA . Two plots are shown: one for the "standard" interaction distance range, and one for small values of r . The small r plot is useful for determining whether the potential is suitable for radiation studies.

The calculation method used is available as the [iprPy diatom_scan](#) calculation method.

Clicking on the image of a plot will open an interactive version of it in a new tab. The underlying data for the plots can be downloaded by clicking on the links above each plot.

Notes and Disclaimers:

- These values are meant to be guidelines for comparing potentials, not the absolute values for any potential's properties. Values listed here may change if the calculation methods are updated due to improvements/corrections. Variations in the values may occur for variations in calculation methods, simulation software and implementations of the interatomic potentials.
- As this calculation only involves two atoms, it neglects any multi-body interactions that may be important in molecules, liquids and crystals.
- [NIST disclaimer](#)

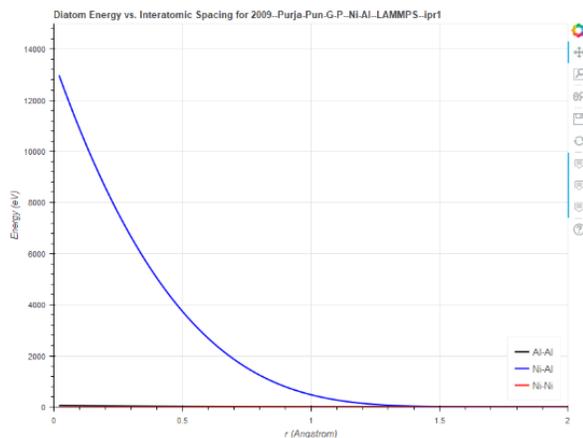
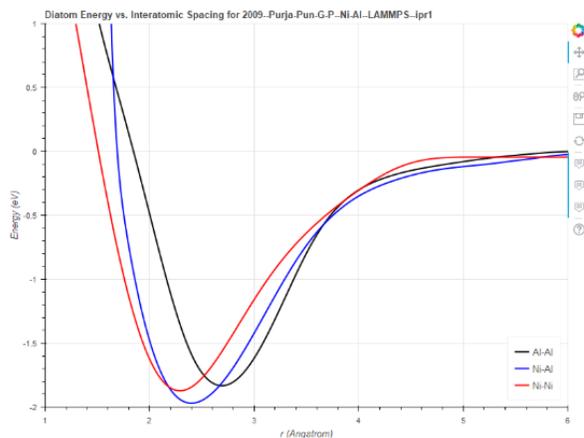
Version Information:

- 2019-11-14. Maximum value range on the shorrange plots are now limited to "expected" levels as details are otherwise lost.
- 2019-08-07. Plots added.

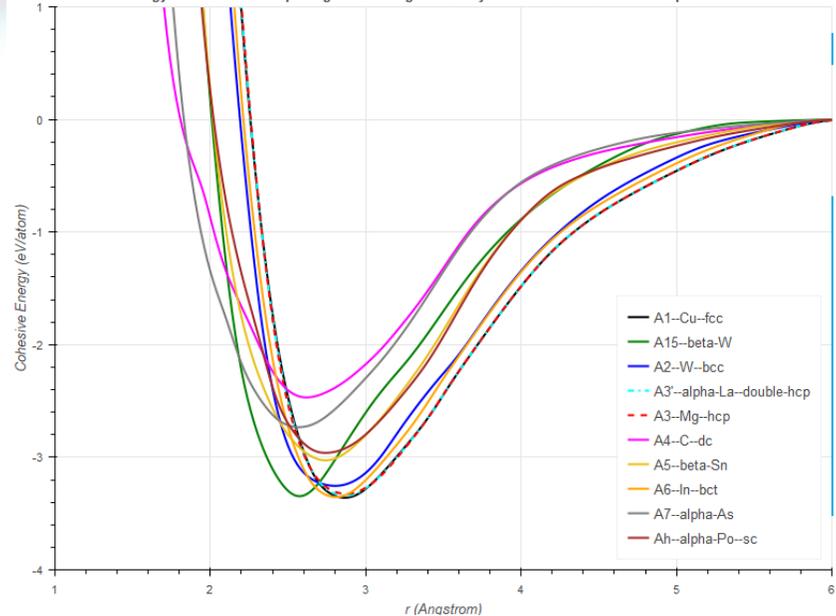
Download data

Click on plot to load interactive version

Click on plot to load interactive version



Cohesive Energy vs. Interatomic Spacing for Al Using 2009-Purja-Pun-G-P-Ni-Al-LAMMPS-IPr1

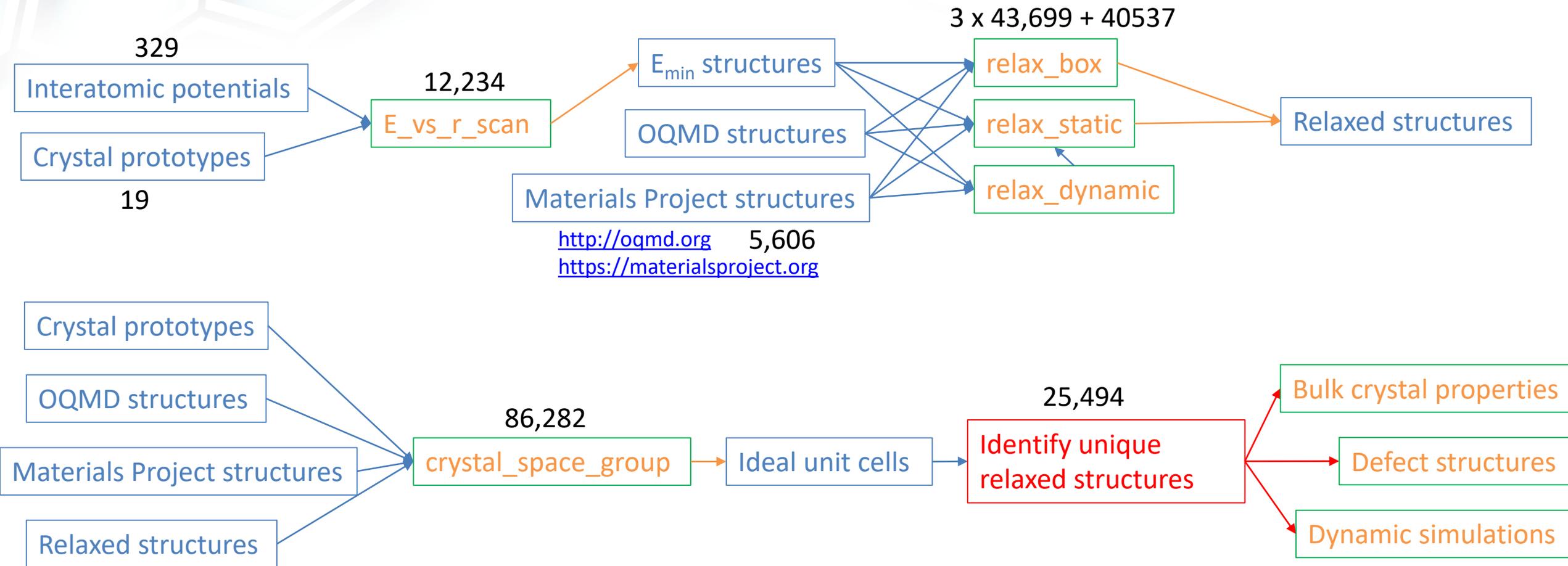


Reference structure matches:

A1--Cu--fcc = mp-23, oqmd-676148
 A15--beta-W = oqmd-1214995
 A2--W--bcc = mp-1008728, oqmd-22516
 A3--alpha-La--double-hcp = oqmd-1215441
 A3--Mg--hcp = mp-10257, oqmd-677944
 A5--beta-Sn = oqmd-1215619
 A6--In--bct = oqmd-1215708

prototype	method	E_{coh} (eV)	a_0 (Å)	b_0 (Å)	c_0 (Å)	α (degrees)	β (degrees)	γ (degrees)
A1--Cu--fcc	dynamic	-4.45	3.52	3.52	3.52	90.0	90.0	90.0
A3--alpha-La--double-hcp	dynamic	-4.4387	2.4854	2.4854	8.1645	90.0	90.0	120.0
oqmd-1216067	static	-4.4351	2.4843	2.4843	18.4039	90.0	90.0	120.0
A3--Mg--hcp	dynamic	-4.4279	2.4819	2.4819	4.1048	90.0	90.0	120.0
A15--beta-W	dynamic	-4.4193	4.4342	4.4342	4.4342	90.0	90.0	90.0
oqmd-1214906	dynamic	-4.3975	6.0662	6.0662	6.0662	90.0	90.0	90.0
oqmd-1214906	box	-4.3971	6.0632	6.0632	6.0632	90.0	90.0	90.0
oqmd-1214817	dynamic	-4.3852	8.626	8.626	8.626	90.0	90.0	90.0
oqmd-1214817	box	-4.3835	8.6317	8.6317	8.6317	90.0	90.0	90.0
A2--W--bcc	static	-4.3827	2.7687	2.7687	2.7687	90.0	90.0	90.0
oqmd-1214728	box	-4.0883	2.4659	2.4659	8.1654	90.0	90.0	90.0
A5--beta-Sn	static	-3.8645	4.6612	4.6612	2.4242	90.0	90.0	90.0
mp-1014111	static	-3.8631	2.4596	2.4596	13.8025	89.1	88.9	60.0
Ah--alpha-Po--sc	box	-3.7264	2.3965	2.3965	2.3965	90.0	90.0	90.0
mp-1094136	box	-3.5827	2.4434	4.2319	15.645	90.0	90.0	90.0
oqmd-1215975	box	-3.4255	3.8699	3.8699	3.9408	90.0	90.0	120.0
A4--C--dc	static	-3.0336	5.2221	5.2221	5.2221	90.0	90.0	90.0

RELAXED CRYSTALS WORKFLOW



PROPERTY CALCULATIONS

Composition:
 Prototype:
 a_0 :
 strain:
[Download raw data](#)

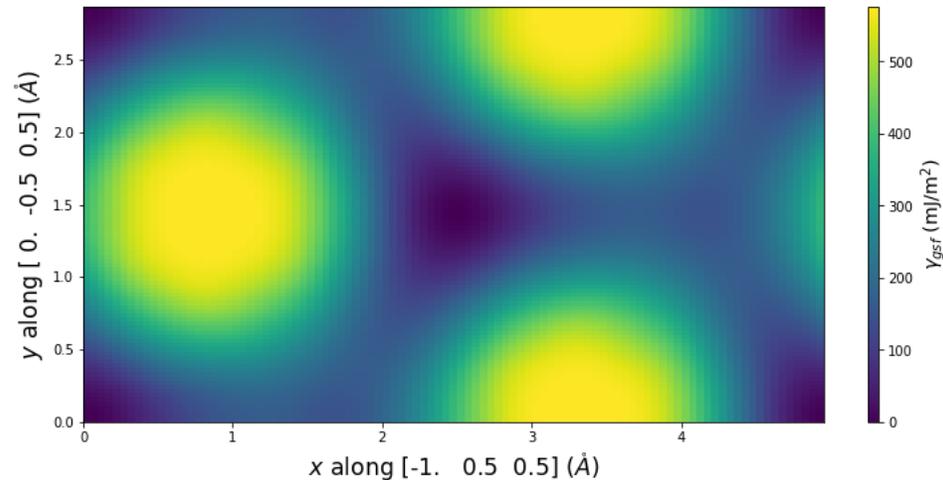
Cij in GPa:

196.217	10.274	28.433	0.0	-0.0	-0.0
10.274	196.217	28.433	0.0	0.0	-0.0
28.433	28.433	208.03	0.0	-0.0	-0.0
-0.0	0.0	-0.0	-47.194	0.0	-0.0
0.0	-0.0	-0.0	0.0	-47.194	-0.0
-0.0	-0.0	-0.0	0.0	0.0	-36.496

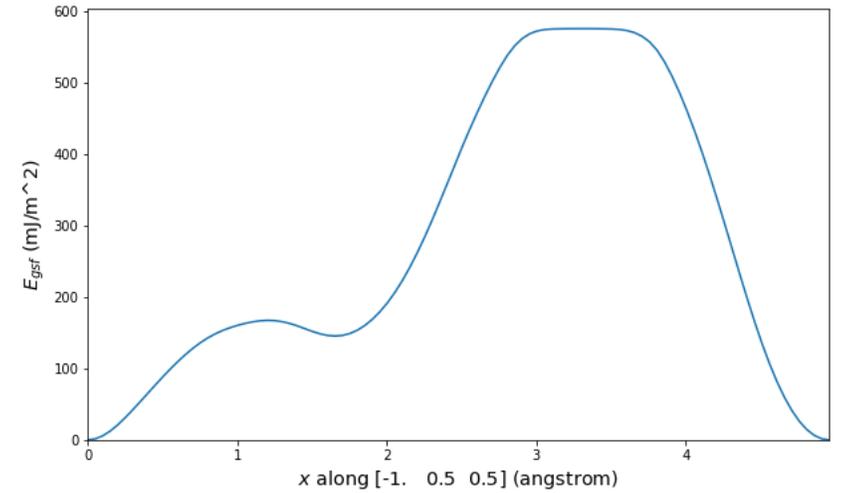
Composition:
 Prototype:
 a_0 :
[Download raw data](#)

Surface	γ_{fs} (mJ/m ²)
(331)	1000.3
(110)	1006.03
(311)	1012.24
(321)	1026.58
(310)	1036.95
(320)	1043.01
(210)	1045.76
(111)	870.52
(100)	943.59
(332)	945.76
(322)	954.7
(221)	976.78
(211)	992.48

Composition:
 Prototype:
 a_0 :
 plot:
[Download raw data](#)



Composition:
 Prototype:
 a_0 :
 plot:
[Download raw data](#)



NEW! RELATED MODELS

Alloy potentials often use previously developed elemental interactions

Identifying families links models and can facilitate further development

“Related models” reveals the potentials that share effectively identical interactions

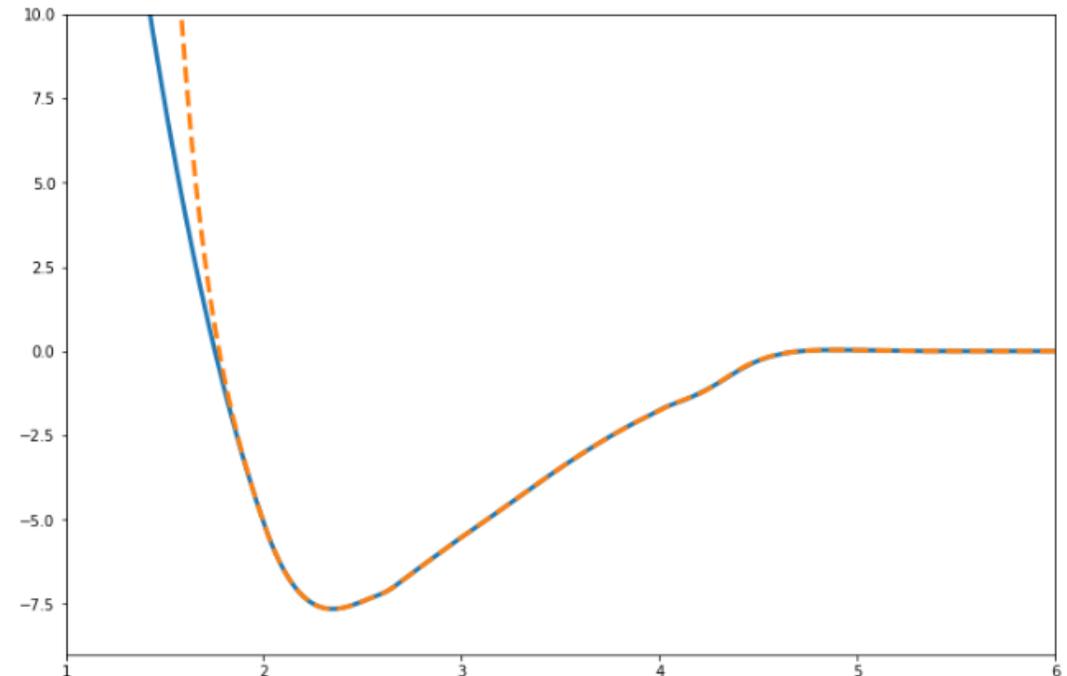
- Matches are identified by comparing diatom energy scans
- “Effective” ranges compared – small r repulsions and cutoffs may differ
- Automatic point-by-point comparisons cuts manual comparisons from 4500 to 50

Related Models:

- [2006--Williams-P-L-Mishin-Y-Hamilton-J-C--Cu-Ag \(Ag\)](#)
- [2009--Wu-H-H-Trinkle-D-R--Cu-Ag \(Ag\)](#)
- [2013--Hale-L-M-Wong-B-M-Zimmerman-J-A-Zhou-X-W--Pd-Ag-H-Hybrid \(Ag\)](#)
- [2013--Hale-L-M-Wong-B-M-Zimmerman-J-A-Zhou-X-W--Pd-Ag-H-Morse \(Ag\)](#)

```
73.80952380952381
2014--Bonny-G--W-H-He-2--LAMMPS--ipr1
2017--Bonny-G--W-Re--LAMMPS--ipr1
```

(-9.0, 10.0)



[HTTPS://POTENTIALS.NIST.GOV](https://potentials.nist.gov)

Contains:

- Potentials listings
- Crystal prototypes
- Relaxed crystals
- Compiled properties
- Defect generation parameters
- Raw calc records (being added)

Explore using keywords or queries

Registered users can contribute

Mendelev *

Search

Tools ▾

From IPR 60

Local Results

- IPR

Federated Search

No federated instance is available.

OAI-PMH

No OAI-PMH Registries available.

Filter by Template [Select All](#)

- Potential
- Family
- FAQ

potential.2003--Mendelev-M-I-Han-S-Srolovitz-D-J-et-al--Fe-2 Potential (Version 1)

potential.2019--Mendelev-M-I--Cu-Zr Potential (Version 1)

potential.2012--Mendelev-M-I-Kramer-M-J-Hao-S-G-et-al--Ni-Zr Potential (Version 1)

potential.2007--Mendelev-M-I-Ackland-G-J--Zr-2 Potential (Version 1)

potential.2007--Mendelev-M-I-Ackland-G-J--Zr-3 Potential (Version 1)

Citation: M.I. Mendelev, and G.J. Ackland (2007), "Development of an interatomic potential for the simulation of phase transformations in zirconium", *Philosophical Magazine Letters*, **87(5)**, 349-359. DOI: [10.1080/09500830701191393](https://doi.org/10.1080/09500830701191393).

Abstract: In recent years, some 30 studies have been published on the molecular dynamics (MD) of zirconium, primarily of its twinning deformation and response to radiation damage. Its low thermal neutron absorption makes it uniquely suited for the latter application. Surprisingly, currently used interatomic potentials do not encapsulate the unique properties of Zr, namely its high stacking-fault energy, anomolous self-diffusion, melting and phase transformation under temperature and pressure (or alloying). Ab initio calculations have shown deficiencies in the description of point defects, both vacancies and interstitials, using existing interatomic potentials, deficiencies that can now be rectified by refitting. Here, we show the calculation of phase transitions

potential.2019--Mendelev-M-I--Fe-Ni-Cr Potential (Version 1)

potential.2015--Berejky-V-Mendelev-M-I-King-A-H-LeSar-B--fictional-Cu-1 Potential (Version 1)

POTENTIALS PYTHON PACKAGE

<https://github.com/usnistgov/potentials>

```
import potentials

potdb = potentials.Database(load='lammps_potentials', verbose=True)

Loaded 276 local LAMMPS potentials
Loaded 276 remote LAMMPS potentials
- 0 new

potdb.lammps_potentials_df
```

	id	key	potid	potkey	units	atom_style	allsymbols	pair_style	status	symbols	elements	masses	charges
0	1985--Foiles-S-M--Ni-Cu--LAMMPS--ipr1	062d2ba7-3903-40ae-a772-daa471d107c6	1985--Foiles-S-M--Ni-Cu	301f04ce-9082-4542-8590-489300cd19e8	metal	atomic	False	eam	active	[Cu, Ni]	[Cu, Ni]	[63.55, 58.71]	[0.0, 0.0]
1	1985--Stillinger-F-H--Si--LAMMPS--ipr1	d085648c-b3ef-4be8-824b-7093fd22770a	1985--Stillinger-F-H-Weber-T-A--Si	edc31ad6-2b9a-455c-9b5f-e888a672ecbd	metal	atomic	False	sw	active	[Si]	[Si]	[28.085]	[0.0]
2	1986--Foiles-S-M--Ag--LAMMPS--ipr1	76a265fc-45ff-49d7-8c64-2044f12402f2	1986--Foiles-S-M-Baskes-M-I-Daw-M-S--Ag	672d54f8-9f48-4200-af56-8a7378ebbc4a	metal	atomic	False	eam	active	[Ag]	[Ag]	[107.87]	[0.0]
3	1986--Foiles-S-M--Ag-Au-Cu-Ni-Pd-Pt--LAMMPS--ipr1	c5afa7e8-6b3b-49cd-ad1c-ae3e4329363a	1986--Foiles-S-M-Baskes-M-I-Daw-M-S--Ag-Au-Cu...	7a1302de-59cf-4efb-900e-cad845b68ee5	metal	atomic	False	eam	active	[Ag, Au, Cu, Ni, Pd, Pt]	[Ag, Au, Cu, Ni, Pd, Pt]	[107.87, 196.97, 63.55, 58.71, 106.4, 195.09]	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
4	1986--Foiles-S-M--Au--LAMMPS--ipr1	c588810a-b96d-4871-bfe2-cff8a5a7c709	1986--Foiles-S-M-Baskes-M-I-Daw-M-S--Au	ffb66faa-319d-4556-8363-dad3959cd553	metal	atomic	False	eam	active	[Au]	[Au]	[196.97]	[0.0]

Local directory

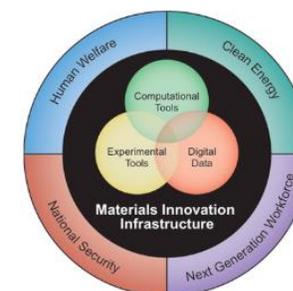
Name	Date modified	Type
crystal_prototype	3/25/2020 3:08 PM	File folder
dislocation	3/25/2020 3:09 PM	File folder
free_surface	3/31/2020 3:03 PM	File folder
point_defect	3/25/2020 3:09 PM	File folder
potential_LAMMPS	7/29/2020 2:45 PM	File folder
reference_crystal	3/25/2020 3:40 PM	File folder
stacking_fault	3/25/2020 3:09 PM	File folder

<https://potentials.nist.gov>

IPR Repository

This system allows for the curation of Material Data in a repository using predefined templates.

This is being developed at the National Institute of Standards and Technology and is made available to solicit comments from the Material Science community. Please do not enter any proprietary data into this system.



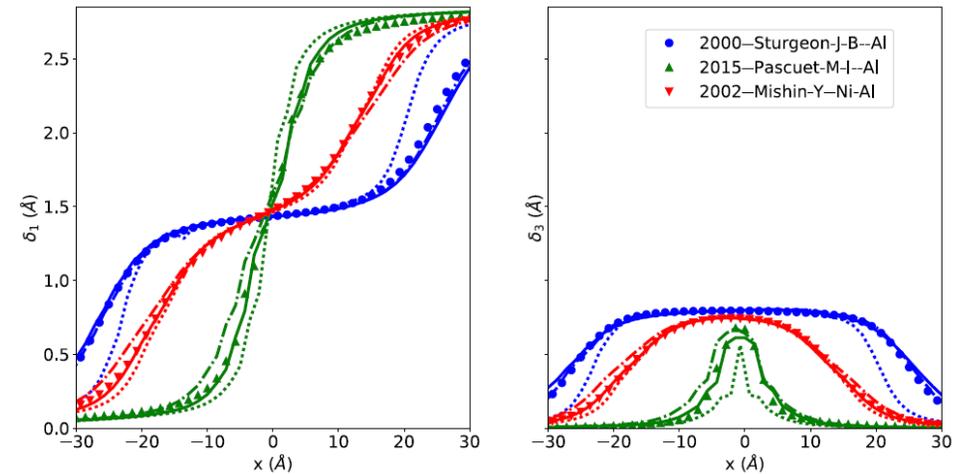
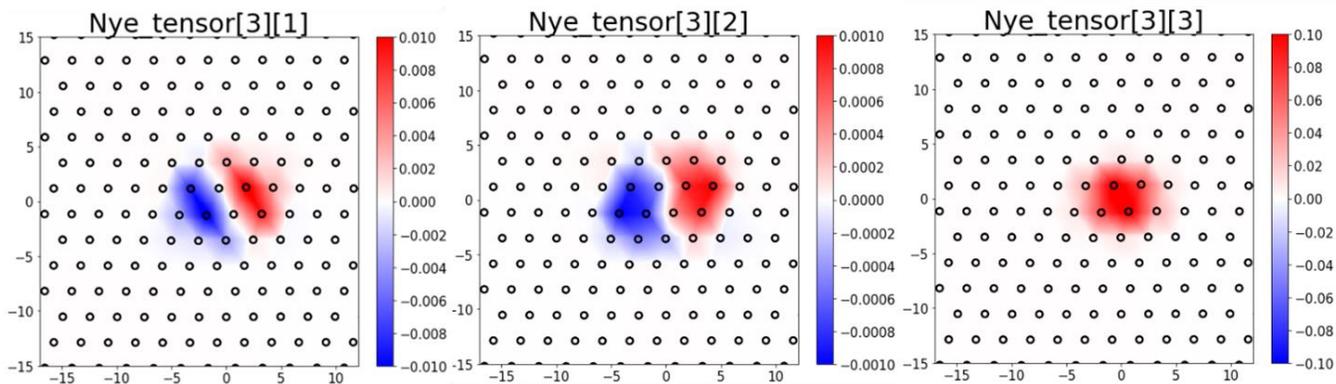
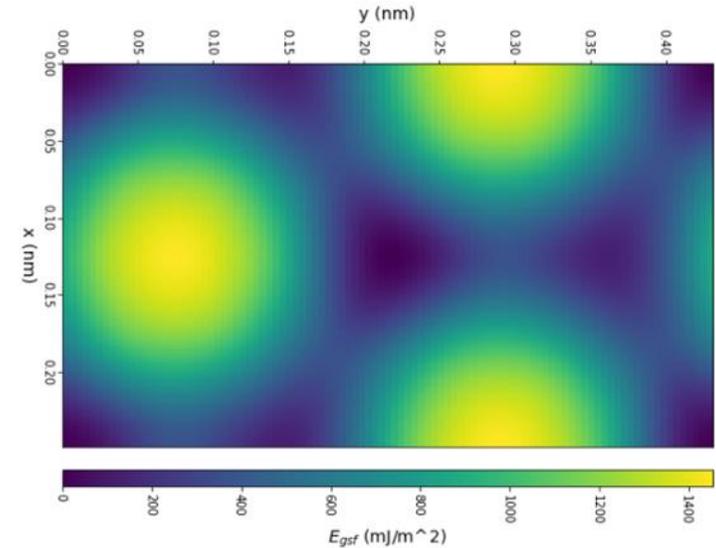
ATOMMAN : ATOMISTIC MANIPULATION TOOLKIT

<https://github.com/usnistgov/atomman>

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

Generic atomic representation designed to support large-scale classical atomistics

- Focus on defect generation + analysis
- Potential/simulator agnostic
- Built-in tools for interacting with LAMMPS



ATOMMAN VS ASE AND PYMATGEN

Atomman was created for classical atomistic calculations rather than DFT

- Atom types, symbols and elements are not the same thing
 - Symbol indicates the interaction model
 - Some potentials have multiple symbols for the same element
 - Some symbols are not for explicit elements
 - Atom types can be assigned same or different symbols
- Atomic systems represent more than cells and molecules
 - Periodic boundary conditions can be changed
 - Cell box has origin position
- Per-atom properties can be freely assigned
 - Can be scalar, vector, or tensor values
 - Not built-in methods – can be calculated in any way
 - Atomic charges may be atom-specific or symbol-specific

But, atomman has built-in converters for `atomman.System` to/from `ase.Atoms`, `pymatgen.Structure`

SOME BASIC ATOMMAN FEATURES

Unit conversions

```
print('5.5 kg/(m*s^2) =')
pressure = uc.set_in_units(5.5, 'kg/(m*s^2)')

print(uc.get_in_units(pressure, 'Pa'), 'Pa')

5.5 kg/(m*s^2) =
5.5 Pa
```

System rotations, multiplications

```
# Rotate system to crystal vectors [110], [-110], [001]
uvws = [[ 1, 1, 0],
        [-1, 1, 0],
        [ 0, 0, 1]]
system = system.rotate(uvws)

# Show system is transformed and expanded
print(system)

avect = [ 5.728,  0.000,  0.000]
bvect = [ 0.000,  5.728,  0.000]
cvect = [ 0.000,  0.000,  4.050]
origin = [ 0.000,  0.000,  0.000]
natoms = 8
natypes = 1
symbols = ('Al',)
pbc = [ True True True]
per-atom properties = ['atype', 'pos']
  id  atype  pos[0]  pos[1]  pos[2]
  0    1    5.728    2.864    4.050
  1    1    0.000    0.000    4.050
  2    1    2.864    5.728    4.050
  3    1    1.432    1.432    2.025
  4    1    1.432    4.296    2.025
  5    1    2.864    2.864    4.050
  6    1    4.296    1.432    2.025
  7    1    4.296    4.296    2.025
```

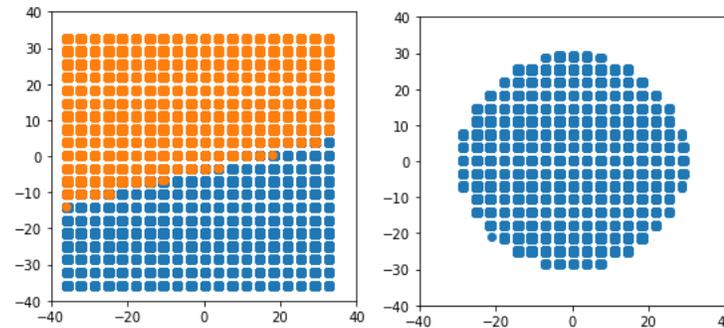
Neighbor lists

```
print('The neighbor atoms of atom 3 are', neighbors[3])
The neighbor atoms of atom 3 are [ 2  4 22 24 202 204 222 224]

# The neighbor lists can also be iterated over for each atom
for nlist in neighbors:
    print(nlist)
    break

[ 1  19 181 199 1801 1819 1981 1999]
```

Geometric region selectors



Miller-Cartesian conversions

```
indices = [[ 2/3, -1/3, -1/3, 0],
           [-1/3, 2/3, -1/3, 0],
           [ 0, 0, 0, 1]]
print(am.tools.miller.vector_crystal_to_cartesian(indices, box))

[[ 2.51    0.    0.    ]
 [-1.255   2.17372376 0.    ]
 [ 0.      0.      4.07   ]]
```

Elastic constants handling

Cij is the 6x6 Voigt representation.

```
[3]: print('Cij (GPa) ->')
      print(uc.get_in_units(C.Cij, 'GPa'))

Cij (GPa) ->
[[114.3  61.9  61.9  0.    0.    0. ]
 [ 61.9 114.3  61.9  0.    0.    0. ]
 [ 61.9  61.9 114.3  0.    0.    0. ]
 [ 0.    0.    0.    31.6  0.    0. ]
 [ 0.    0.    0.    0.    31.6  0. ]
 [ 0.    0.    0.    0.    0.    31.6]]
```

Cij9 is the full 9x9 representation.

```
[4]: print('Cij9 (GPa) ->')
      print(uc.get_in_units(C.Cij9, 'GPa'))

Cij9 (GPa) ->
[[114.3  61.9  61.9  0.    0.    0.    0.    0.    0. ]
 [ 61.9 114.3  61.9  0.    0.    0.    0.    0.    0. ]
 [ 61.9  61.9 114.3  0.    0.    0.    0.    0.    0. ]
 [ 0.    0.    0.    31.6  0.    0.    31.6  0.    0. ]
 [ 0.    0.    0.    0.    31.6  0.    0.    31.6  0. ]
 [ 0.    0.    0.    0.    0.    31.6  0.    0.    31.6 ]
 [ 0.    0.    0.    31.6  0.    0.    31.6  0.    0. ]
 [ 0.    0.    0.    0.    31.6  0.    0.    31.6  0. ]
 [ 0.    0.    0.    0.    0.    31.6  0.    0.    31.6]]
```

DEFECT GENERATION AND ANALYSIS

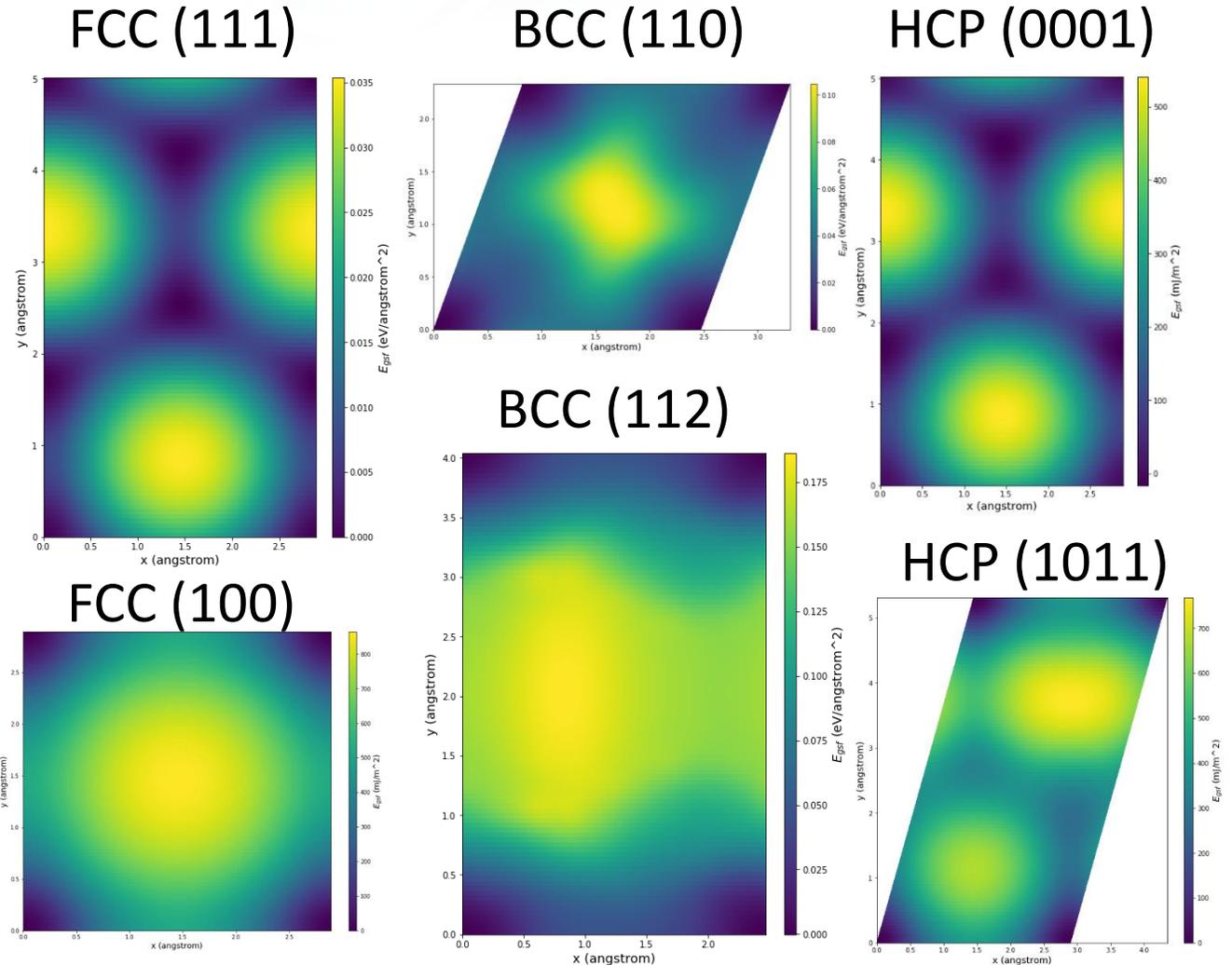
Atomic configuration generators for

- Free surfaces
- Stacking faults
- Point defects
- Dislocations

Gamma surface class – point, 1D, 2D plots

Dislocation structure analysis methods

- Disregistry
- Differential displacement
- Slip vector
- Nye tensor
- Semi-discrete Variational Peierls-Nabarro



POTENTIALS.NIST.GOV DATABASE ACCESSING

NEW! Atomman now uses potentials package – load potentials, prototypes and structures from database

Load potential and relaxed crystal structure from database

```
potential = am.load_lammps_potential(id='2006--Williams-P-L--Ag--LAMMPS--ipr1')
print(potential)
```

```
2006--Williams-P-L--Ag--LAMMPS--ipr1
```

```
ucell = am.load('crystal', potential=potential, family='A1--Cu--fcc')
print(ucell)
```

```
avect = [ 4.090,  0.000,  0.000]
bvect = [ 0.000,  4.090,  0.000]
cvect = [ 0.000,  0.000,  4.090]
origin = [ 0.000,  0.000,  0.000]
natoms = 4
natypes = 1
symbols = ('Ag',)
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 | 2.045 | 2.045
  2 | 1 | 2.045 | 0.000 | 2.045
  3 | 1 | 2.045 | 2.045 | 0.000
```

IPRPY CALCULATION FRAMEWORK

Property calculation methods with high-throughput tools

- source: <https://github.com/usnistgov/iprPy>
- docs: <https://www.ctcms.nist.gov/potentials/iprPy>

Make calculation methods as accessible as possible

- Openly available
- Low barrier for usage at all levels
- Transparent, documented methodologies
- Adaptable to new materials
- Transferable to other frameworks

Need data, but also need to *trust* the data!

jupyter stacking_fault_map_2D Last Checkpoint: 07/30/2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

stacking_fault_map_2D calculation style

Lucas M. Hale, lucas.hale@nist.gov, Materials Science and Engineering Division, NIST.

Description updated: 2019-07-26

Introduction

The `stacking_fault_map_2D` calculation style evaluates the full 2D generalized stacking fault map for an array of shifts along a specified crystallographic plane. A regular grid of points is established and the generalized stacking fault energy is evaluated at each.

Version notes

- This was previously called the `stacking_fault_multi` calculation and was renamed for clarity.

Additional dependencies

Disclaimers

- [NIST disclaimers](#)
- The system's dimension perpendicular to the fault plane should be large to minimize the interaction of the free surface and the stacking fault.

Method and Theory

First, an initial system is generated. This is accomplished using `atomman.defect.StackingFault`, which

1. Starts with a unit cell system.
2. Generates a transformed system by rotating the unit cell such that the new system's box vectors correspond to crystallographic directions, and filled in with atoms to remain a perfect bulk cell when the three boundaries are periodic.
3. All atoms are shifted by a fractional amount of the box vectors if needed.
4. A supercell system is constructed by combining multiple replicas of the transformed system.
5. The system is then cut by making one of the box boundaries non-periodic. A limitation placed on the calculation is that the normal to the cut plane must correspond to one of the three Cartesian (x , y , or z) axes. If true, then of the system's three box vectors (\vec{a} , \vec{b} , and \vec{c}), two will be parallel to the plane, and the third will not. The non-parallel box vector is called the cutboxvector, and for LAMMPS compatible systems, the following conditions can be used to check the system's compatibility:
 - `cutboxvector = 'c'`: all systems allowed.
 - `cutboxvector = 'b'`: the system's yz tilt must be zero.
 - `cutboxvector = 'a'`: the system's xy and xz tilts must be zero.

A LAMMPS simulation performs an energy/force minimization on the system where the atoms are confined to only relax along the Cartesian direction normal to the cut plane.

A mathematical fault plane parallel to the cut plane is defined in the middle of the system. A generalized stacking fault system can then be created by shifting all atoms on one side of the fault plane by a vector \vec{s} . The shifted system is then relaxed using the same confined energy/force minimization used on the non-

IMPLEMENTED CALCULATIONS

Python script: “black box” run from simple text input script

Python class: directly call underlying calculation methods

Jupyter Notebook: instructional “clear box” guide with description, code and example

Workflows: each is an independent unit of work

```
# Input script for calc_E_vs_r_scan.py

# Command lines for LAMMPS and MPI
lammmps_command      lmp_serial
mpi_command

# Potential definition and directory containing associated files
potential_file        1989--Adams-J-B--Ag--LAMMPS--ipr1.json
potential_dir         1989--Adams-J-B--Ag--LAMMPS--ipr1

# Initial system configuration to load
load_file             Al--Cu--fcc.json
load_style            system_model
load_options
family
symbols              Ag
box_parameters

# System manipulations
a_uvw
b_uvw
c_uvw
atomshift
sizemults           5 5 5

# Units for input/output values
length_unit
pressure_unit
energy_unit
force_unit

# Run parameters
minimum_r            0.5
maximum_r            6
number_of_steps_r   276
```

IMPLEMENTED CALCULATIONS

Python script: “black box” run from simple text input script

Python class: directly call underlying calculation methods

Jupyter Notebook: instructional “clear box” guide with description, code and example

Workflows: each is an independent unit of work

```
import atomman as am
import iprPy

cohesive_scan = iprPy.load_calculation('E_vs_r_scan')

lammmps_command = 'lmp_mpi'

ucell = am.load('system_model', 'Al--Cu--fcc.json', symbols = 'Ag')
system = ucell.supersize(5, 5, 5)

potential = am.lammps.Potential('1989--Adams-J-B--Ag--LAMMPS--ipr1.json')

rmin = 0.5
rmax = 6
rsteps = 276

cohesive_scan.calc(lammmps_command, system, potential, ucell, rmin, rmax, rsteps)
```

IMPLEMENTED CALCULATIONS

Python script: “black box” run from simple text input script

Python class: directly call underlying calculation methods

Jupyter Notebook: instructional “clear box” guide with description, code and example

Workflows: each is an independent unit of work

jupyter E_vs_r_scan Last Checkpoint: 09/25/2018 (autosaved) Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted

E_vs_r_scan Calculation

Lucas M. Hale, lucas.hale@nist.gov, Materials Science and Engineering Division, NIST.
Chandler A. Becker, chandler.becker@nist.gov, Office of Data and Informatics, NIST.
Zachary T. Trautt, zachary.trautt@nist.gov, Materials Measurement Science Division, NIST.

Version: 2018-06-24

[Disclaimers](#)

Introduction

The E_vs_r_scan calculation creates a plot of the cohesive energy vs interatomic spacing, r , for a given atomic system. The system size is uniformly scaled (b/a and c/a ratios held fixed) and the energy is calculated at a number of sizes without relaxing the system. All box sizes corresponding to energy minima are identified.

This calculation was created as a quick method for scanning the phase space of a crystal structure with a given potential in order to identify starting guesses for further structure refinement calculations.

Disclaimer #1: the minima identified by this calculation do not guarantee that the associated crystal structure will be stable as no relaxation is performed by this calculation. Upon relaxation, the atomic positions and box dimensions may transform the system to a different structure

Disclaimer #2: it is possible that the calculation may miss an existing minima for a crystal structure if it is outside the range of r values scanned, or has b/a , c/a values far from the ideal.

Method and Theory

An initial system (and corresponding unit cell system) is supplied. The r/a ratio is identified from the unit cell. The system is then uniformly scaled to all r_i values in the range to be explored and the energy for each is evaluated using LAMMPS and “run 0” command, i.e. no relaxations are performed.

In identifying energy minima along the curve, only the explored values are used without interpolation. In this way, the possible energy minima structures are identified for r_i where $E(r_i) < E(r_{i-1})$ and $E(r_i) < E(r_{i+1})$.

Demonstration

MODULAR METHOD DEVELOPMENT

Calculations, Records, Databases all modular subclasses – independent requirements

Modules for common inputs – faster development

```
# Input script for calc_E_vs_r_scan.py

# Command lines for LAMMPS and MPI
lammps_command ..... lmp_mpi
mpi_command .....

# Potential definition and directory containing associated files
potential_file ..... 2004--Zhou-X-W--Pt--LAMMPS--ipr2.json
potential_dir ..... 2004--Zhou-X-W--Pt--LAMMPS--ipr2

# Initial system configuration to load
load_file ..... Al--Cu--fcc.json
load_style ..... system_model
load_options .....
family ..... Al--Cu--fcc
symbols ..... Pt
box_parameters .....

# System manipulations
a_uvw .....
b_uvw .....
c_uvw .....
atomshift .....
sizemults ..... 10 10 10

# Units for input/output values
length_unit .....
pressure_unit .....
energy_unit .....
force_unit .....

# Run parameters
minimum_r ..... 0.5
maximum_r ..... 6.0
number_of_steps_r ..... 276

# Check lammps_command and mpi_command
iprPy.input.interpret('lammps_commands', input_dict)

# Load potential
iprPy.input.interpret('lammps_potential', input_dict)

# Load ucell system
iprPy.input.interpret('atomman_systemload', input_dict, build=build)

# Add atomic charges
iprPy.input.interpret('lammps_atomcharges', input_dict, build=build)

# Construct initials system by manipulating ucell system
iprPy.input.interpret('atomman_systemmanipulate', input_dict, build=build)

# Run e_vs_r
results_dict = e_vs_r(input_dict['lammps_command'],
                     input_dict['initialsystem'],
                     input_dict['potential'],
                     mpi_command = input_dict['mpi_command'],
                     ucell = input_dict['ucell'],
                     rmin = input_dict['minimum_r'],
                     rmax = input_dict['maximum_r'],
                     rsteps = input_dict['number_of_steps_r'])

"calculation-E-vs-r-scan": {
  "key": "036f4150-3936-48c6-8b8d-f0cce6e2dc47",
  "calculation": {
    "iprPy-version": "0.8.3",
    "atomman-version": "1.2.3",
    "LAMMPS-version": "5.Sep.2018",
    "script": "calc_E_vs_r_scan",
    "run-parameter": {
      "size-multipliers": {
        "a": [0, 10],
        "b": [0, 10],
        "c": [0, 10]},
      "minimum_r": {
        "value": 0.5,
        "unit": "angstrom"},
      "maximum_r": {
        "value": 6.0,
        "unit": "angstrom"},
      "number_of_steps_r": 276},
    "potential-LAMMPS": {
      "key": "94afffa5-6a8d-4a97-a178-1412e65a2ffc",
      "id": "2004--Zhou-X-W--Pt--LAMMPS--ipr2",
      "potential": {
        "key": "b236a7be-023c-4df7-8146-da4fbd9cf900",
        "id": "2004--Zhou-X-W--Johnson-R-A-Wadley-H-N-G--Pt"}},
    "system-info": {
      "family": "Al--Cu--fcc",
      "artifact": {
        "file": "Al--Cu--fcc.json",
        "format": "system_model",
        "load_options": null},
      "symbol": "Pt"},
  }
}
```

PACKAGED CALCULATIONS (IN PROGRESS)

Pip install iprPy (conda coming soon)
GUI inputs (ipywidgets, nanoHUB, more?)
“Black box” and “clear box” versions

3. Calculation Input Parameters

- LAMMPS: LAMMPS executable to use
- Potential: Name of the interatomic potential to use
- Symbol 1 and 2: The elemental models from the potential to use for the two atoms
- Min and Max r: The range of interatomic distances to explore
- Num r steps: The number of measurements to make

```
: input_dict = {}  
  widget_diatom_scan(input_dict)
```

LAMMPS:

Potential:

Symbol 1:

Symbol 2:

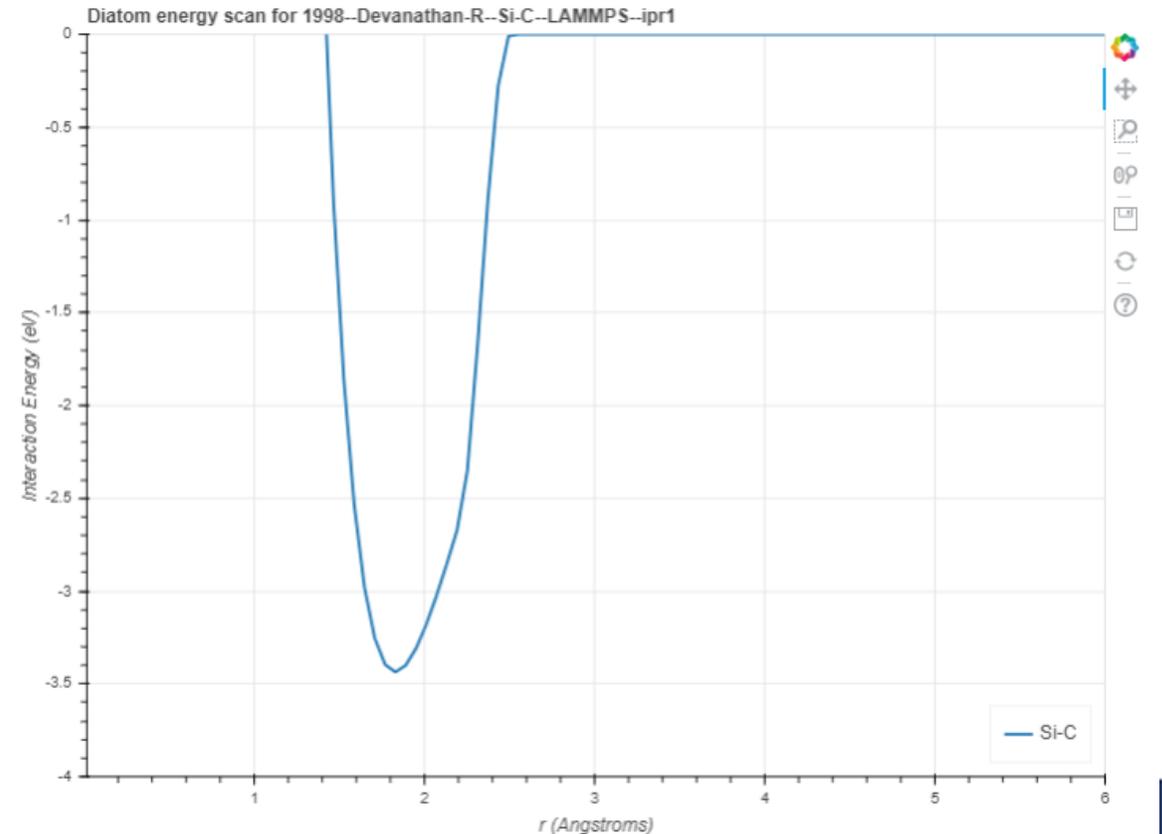
Min r:

Max r:

Num r steps:

4. Run calculation function(s) and display results

```
: results_dict = diatom(input_dict['lammps_command'],  
                        input_dict['potential'],  
                        input_dict['symbols'],  
                        rmin = input_dict['rmin'],  
                        rmax = input_dict['rmax'],  
                        rsteps = input_dict['rsteps'])  
  
show_E_vs_r_plot(input_dict, results_dict)
```



JARVIS

<https://www.ctcms.nist.gov/~knc6/JARVIS.html>

JARVIS DFT

DFT database with
~40,000 3D materials
~1,000 2D materials

Primarily uses vdW-DF-OptB88
functional, and uses other beyond-
GGA methods when needed

Conventional and unique datasets of
property measurements

automatic k-point convergence
protocol



JARVIS FF

Database and framework for direct
comparisons of classical potential
properties to corresponding DFT results

JARVIS ML

ML models based on DFT results to
predict properties for “new”
structures

Used for fast materials-screening and
energy-landscape mapping

These models, the workflow, dataset
etc. are disseminated to enhance the
transparency of the work

[JARVIS: An Integrated Infrastructure for Data-driven Materials Design, arXiv:2007.01831 \(2020\).](#)

MATERIALS RESOURCE REGISTRY

<https://materials.registry.nist.gov/>

The screenshot shows the homepage of the Materials Resource Registry. At the top left is the NIST logo. To the right are navigation links: Home, Services, Login, Help, and Contact. The main heading is "Materials Resource Registry" with the subtitle "Part of the Materials Genome Initiative". Below this are two buttons: "SEARCH FOR RESOURCES" and "ADD YOUR RESOURCE".

Find Materials Data

This system allows for the registration of materials resources, bridging the gap between existing resources and the end users. The Materials Resource Registry functions as a centrally located service, making the registered information available for research to the materials community.

This is being developed at the National Institute of Standards and Technology and is made available to

A vertical sidebar menu with a blue header "Home Page". Below it are three items: "Services", "Search for resources", and "Add your resource".

The screenshot shows search results for "Polymer Property Predictor and Database". The browser address bar shows "materials.registry.nist.gov" and "261 results". The search criteria used are "Type". The results are categorized by "TYPE", "ORIGIN OF DATA", "MATERIAL TYPE", and "STRUCTURAL FEATURE".

- Polymer Property Predictor and Database** (University of Chicago)
<http://pppdb.uchicago.edu/>
The Polymer Property Predictor and Database includes both a database of polymer interaction parameters (χ), glass transition temperatures, as well as tools to predict polymer properties and phase diagrams. Phase diagrams for both neutral polymers (Flory-Huggins and Lattice Cluster Theory) and charged polymers (Voorn-Overbeek) can be generated give... [show more](#)
- ZENO** (Jack Douglas - NIST)
<https://github.com/usnistgov/zeno>
Subject keyword(s): Monte Carlo, Stokes friction coefficient, Electrostatic capacity, Intrinsic viscosity, Intrinsic conductivity, Electrical polarizability
- National Institute for Computational Sciences, Oak Ridge National Laboratory** (University of Tennessee)
<https://www.nics.tennessee.edu/>
Subject keyword(s): high performance computing, large-scale data analysis, data visualization, XSEDE
The National Institute for Computational Sciences (NICS) at the University of Tennessee, Knoxville is one of the leading high performance computing centers for excellence in the United States. NICS strives to accomplish [its] mission by facilitating transformational scientific discovery by providing scientists and researchers from around ... [show more](#)
- Potfit** (Peter Brommer, Franz Gähler - Potfit)

Red arrows point to specific elements: "See detailed metadata" points to the "show more" link in the first result; "Visit resource's home page" points to the URL in the first result; "Read full description" points to the "show more" link in the second result.

LINKS

Interatomic Potentials Repository

<https://www.ctcms.nist.gov/potentials/>

Potentials database

<https://potentials.nist.gov>

Python potentials

<https://github.com/usnistgov/potentials> (stable)

<https://github.com/lmhale99/potentials> (development)

Python atomman

<https://www.ctcms.nist.gov/potentials/atomman> (documentation)

<https://github.com/usnistgov/atomman> (stable)

<https://github.com/usnistgov/atomman> (development)

Python iprPy

<https://www.ctcms.nist.gov/potentials/iprPy> (documentation)

<https://github.com/usnistgov/iprPy> (stable)

<https://github.com/usnistgov/iprPy> (development)

potentials, atomman and iprPy can be installed from code, pip, or conda-forge

Demonstrations of atomman and iprPy tomorrow at 10 AM