







Artificial Intelligence

Can a Large Language Model replace a scientist?



Artificial Intelligence

Can a Large Language Model replace a scientist?



Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.



Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.

Physical Parameters,



Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.

ע

Physical Parameters, Technical Parameters

Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.



Answer form ChatGPT:

Physical Parameters, Technical Parameters



To calculate the bulk modulus of aluminum using the Atomic Simulation Environment (ASE) with the Effective Medium Theory (EMT) potential, you can follow these steps:

- 1. Set up the aluminum bulk structure.
- 2. Assign the EMT calculator to the structure.
- 3. Compute the energies at different volumes by scaling the unit cell.
- 4. Fit the calculated energies and volumes to an equation of state (EOS).
- 5. Extract the bulk modulus from the EOS fit.

Hands-on: Let's give it a try!

Answer form ChatGPT:

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.



[2]: # Create an aluminum bulk structure al_bulk = bulk('Al', 'fcc', a=4.05, cubic=True)

```
[3]: # Attach the EMT calculator
al_bulk.calc = EMT()
```

[4]: # Define a range of scaling factors for volume scaling_factors = np.linspace(0.94, 1.06, 10) # Scale between 94% and 106% volumes = [] energies = []



Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.

Answer form ChatGPT:



Hands-on: Let's give it a try!

Answer form ChatGPT:

Calculate the bulk modulus for aluminium with ASE using

the EMT simulation code.



volumes.append(al_bulk.get_volume()) energies.append(al bulk.get potential energy())

[6]: # Fit the Equation of State (EOS) eos = EquationOfState(volumes, energies) v0, e0, B = eos.fit() # Fit returns equilibrium volume, energy, and bulk modulus

[7]: # Print results print(f"Equilibrium volume: {v0:.2f} Å³") print(f"Equilibrium energy: {e0:.2f} eV") print(f"Bulk modulus: {B:.2f} GPa")

> Equilibrium volume: 62.50 Å³ Equilibrium energy: -0.03 eV Bulk modulus: 0.26 GPa

Hands-on: Let's give it a try!

Answer form ChatGPT:

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.



```
[6]: # Fit the Equation of State (EOS)
eos = EquationOfState(volumes, energies)
v0, e0, B = eos.fit() # Fit returns equilibrium volume, energy, and bulk modulus
```

[7]: # Print results
print(f"Equilibrium volume: {v0:.2f} Å³")
print(f"Equilibrium energy: {e0:.2f} eV")
print(f"Bulk modulus: {B:.2f} GPa")

Equilibrium volume: 62.50 Å³ Equilibrium energy: -0.03 eV Bulk modulus: 0.26 GPa





Hands-on: Let's give it a try!

Answer form ChatGPT:

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.



Print results
print(f"Equilibrium volume: {v0:.2f} Å³")
print(f"Equilibrium energy: {e0:.2f} eV")
print(f"Bulk modulus: {B:.2f} GPa")

Equilibrium volume: 62.50 Å³ Equilibrium energy: -0.03 eV Bulk modulus: 0.26 GPa

Experiment: 62 GPa



Hands-on: Let's give it a try!

Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.





With Correction: 40 GPa Experiment: 62 GPa

MPI for Sustainable Materials | Jan Janssen

https://chatgpt.com



Hands-on: Let's give it a try!



Calculate the bulk modulus for aluminium with ASE using the EMT simulation code.

Debugging:

eos.fit??

Signature: eos.fit(warn=True) def fit(self, warn=True):

The code produced by ChatGPT 40 can be executed. The generated code is >90% correct, but the scientific result is wrong.

```
print(B / kJ * 1.0e24, 'GPa')
```

With Correction: 40 GPa Experiment: 62 GPa

MPI for Sustainable Materials | Jan Janssen

https://chatgpt.com

Materials Informatics Group



Our Expertise: Workflows for Sustainable Materials



Materials Informatics Group



Our Expertise: Workflows for Sustainable Materials



MPI for Sustainable Materials | Jan Janssen J. Janssen, et al., Comp. Mat. Sci. 161 (2019) - https://pyiron.org

Materials Informatics Group



Our Expertise: Workflows for Sustainable Materials



MPI for Sustainable Materials | Jan Janssen J. Janssen, et al., Comp. Mat. Sci. 161 (2019) - https://pyiron.org



pyiron based Large Language Model Interface for Atomistic Simulation

8	+	Ж		Ċ		-	G	••	Code	~	Notebook 🗋	Ŭ	Python 3 (ipyker	nel) 🔿 🔳
	[1	1:	%re	load	_ext	lar	ngsi	m						
8) [2]:	%%cl I ai Can	hat m lo you	okin com	g f a	or a e th	mat e bu	erial w Ik modu	ith a bulk lus for th	c modulus in the ne noble metals	rar usir	nge of 200 GPa. ng EMT for me?	
\$			Her	e are	the	com	nput	ed b	ulk modu	li for the no	ble metals using t	he E	MT model:	
			•	Gole	d (Aı	u) : 1	73.8	6 GF	Pa					
			•	Silv	er (A	\g) :	100.	15 G	Pa					

• Platinum (Pt): 278.20 GPa

Among these, Platinum (Pt) has a bulk modulus within the range of 200 GPa.



pyiron based Large Language Model Interface for Atomistic Simulation

6 + %	□ □ ► ■ C ► Code Notebook □ Python 3 (ipykernel) ○ ■
[1]:	%reload_ext langsim
(2) ^[2] :	‰chat I am looking for a material with a bulk modulus in the range of 200 GPa. Can you compute the bulk modulus for the noble metals using EMT for me?
S	 Here are the computed bulk moduli for the noble metals using the EMT model: Gold (Au): 173.86 GPa Silver (Ag): 100.15 GPa Platinum (Pt): 278.20 GPa Among these, Platinum (Pt) has a bulk modulus within the range of 200 GPa.
(3]:	‰chat How does this compare to experiment?
S	 Here are the experimental bulk moduli for the noble metals: Gold (Au): 180 GPa Silver (Ag): 100 GPa Platinum (Pt): 230 GPa



pyiron based Large Language Model Interface for Atomistic Simulation

8	+	Ж	Ū	Ċ			C	••	Code	\checkmark		Noteb	book 🗋	Ŭ	Python	3 (ip	ykernel)	○ ■
	[1	1]:	%re	load _.	_ext	lar	ngsi	m										
8) [2	2]:	%%c I ai Can	hat m lo you	okin com	g fa pute	or a e th	mat e bu	erial wit Ik modulu	h a bul s for t	lk moo the no	dulus oble m	in the etals	ran usin	ge of 2 g EMT 1	200 (for r	GPa. ne?	
\$																		

> Entering new AgentExecutor chain...

```
Invoking: `get_atom_dict_bulk_structure` with `{'chemical_symbol': 'Au'}`
numbers=[79] positions=[[0.0, 0.0, 0.0]] cell=[[0.0, 2.04, 2.04], [2.04, 0.0,
2.04], [2.04, 2.04, 0.0]] pbc=[True, True, True]
Invoking: `get_atom_dict_equilibrated_structure` with `{'atom_dict': {'number
s': [79], 'positions': [[0.0, 0.0, 0.0]], 'cell': [[0.0, 2.04, 2.04], [2.04, 0.
0, 2.04], [2.04, 2.04, 0.0]], 'pbc': [True, True, True]}, 'calculator_str': 'em
t'}`
```

	Step	Time	Energy	fmax
LBFGS:	0	19:55:11	0.002606	0.308859
LBFGS:	1	19:55:11	0.000032	0.077808
LBFGS:	2	19:55:11	-0.000135	0.003099
LBFGS:	3	19:55:11	-0.000135	0.000029
LBFGS:	4	19:55:11	-0.000135	0.00000



pyiron based Large Language Model Interface for Atomistic Simulation

8	+	Ж		Ċ		•	C	••	Code	\checkmark		Note	book 🗋	Ŭ	Pyth	on 3 ((ipyke	rnel) 🔘	
	[1	1]:	%re	load _.	_ext	: laı	ngsi	m											
8) [2	2]:	%%cl I ai Can	hat m lo you	okin com	ig f o iputo	or a e th	mat e bu	erial wi lk modul	th a bul us for t	lk mod the no	lulus ble m	in the metals	e rar usir	ige of ig EM ⁻	f 200 F for	0 GPa r me?		
\$																			

> Entering new AgentExecutor chain...

Invoking numbers=	g: `get_atom_dic =[79] positions=	t_bulk_structure` [[0.0, 0.0, 0.0]]	<pre>with `{'chemical_ cell=[[0.0, 2.04, e Truel</pre>	symbol': 'Au'}` 2.04], [2.04, 0.0,
Invoking s': [79] 0, 2.04] t'}`	; `get_atom_dic , 'positions': , [2.04, 2.04, 0	t_equilibrated_st [[0.0, 0.0, 0.0]] 0.0]], 'pbc': [Tr	ructure` with `{'at , 'cell': [[0.0, 2] ue, True, True]},	tom_dict': {'number 04, 2.04], [2.04, 0. 'calculator_str': 'em
S	Step Time	Energy	fmax	
LBFGS:	0 19:55:11	0.002606	0.308859	
LBFGS:	1 19:55:11	0.000032	0.077808	
LBFGS:	2 19:55:11	-0.000135	0.003099	
LBFGS:	3 19:55:11	-0.000135	0.000029	
LBFGS:	4 19:55:11	-0.000135	0.00000	





pyiron based Large Language Model Interface for Atomistic Simulation

	+ %	□ □ ► ■ C ► Code ➤ Notebook □ Python 3 (ipykernel) ○ ■
		%reload_ext langsim
\otimes		%%chat I am looking for a material with a bulk modulus in the range of 200 GPa. Can you compute the bulk modulus for the noble metals using EMT for me?

Using specialized agents, the Large Language Model (LLM) can interface with atomistic simulation codes and utilities.

2.04], Invoking s': [79] 0, 2.04] t'}`	[2.04, 2.04, 0 g: `get_atom_d:], 'positions'], [2.04, 2.04	.0]] pbc=[True, T ict_equilibrated_: : [[0.0, 0.0, 0.0 , 0.0]], 'pbc': [`	rue, True] structure` with `{'at]], 'cell': [[0.0, 2. True, True, True]}, '	om_dict': {'number 04, 2.04], [2.04, 0. calculator_str': 'em
9	Step Time	Energy	fmax	
LBFGS:	0 19:55:11	0.002606	0.308859	
LBFGS:	1 19:55:11	0.000032	0.077808	
LBFGS:	2 19:55:11	-0.000135	0.003099	
LBFGS:	3 19:55:11	-0.000135	0.000029	
LBFGS:	4 19:55:11	-0.000135	0 . 00000	





Compare Scientific Software Installation to Downloading an App on your Smartphone

How to get the code?

- Get the source code
- Install the required dependencies
- Compile the software



Compare Scientific Software Installation to Downloading an App on your Smartphone

How to get the code?

- · Get the source code
- Install the required dependencies
- Compile the software

How to make VASP

Copy one of the makefile.include.* files in root/arch to root/makefile.include. Take one that most closely reflects your system (hopefully).

cp arch/makefile.include.linux_intel ./makefile.include

In many cases these makefile.include files will have to be adapted to the particulars of your system (see below).

When you've finished setting up makefile.include , build VASP:

make all



Compare Scientific Software Installation to Downloading an App on your Smartphone

How to get the code?

- Get the source code
- Install the required dependencies
- Compile the software

How to make VASP

Copy one of the makefile.include.* files in root/arch to root/makefile.include. Take one that most closely reflects your system (hopefully).

cp arch/makefile.include.linux_intel ./makefile.include

In many cases these makefile.include files will have to be adapted to the particulars of your system (see below).

When you've finished setting up makefile.include, build VASP:

make all



Compare Scientific Software Installation to Downloading an App on your Smartphone

How to get the code?

- · Get the source code
- Install the required dependencies
- Compile the software

How to make VASP

Copy one of the makefile.include.* files in root/arch to root/makefile.include. Take one that most closely reflects your system (hopefully).

cp arch/makefile.include.linux_intel ./makefile.include

In many cases these makefile.include files will have to be adapted to the particulars of your system (see below).

When you've finished setting up makefile.include , build VASP:

make all

Installation Documentation

- VASP 19 pages
- LAMMPS 50 pages
- DAMASK 14 pages



Compare Scientific Software Installation to Downloading an App on your Smartphone

How to get the code?

- Get the source code
- Install the required dependencies
- Compile the software

Installation Documentation

- VASP 19 pages
- LAMMPS 50 pages
- DAMASK 14 pages

While the source code is available, the installation of a scientific software can be challenging.

In many cases these makefile.include files will have to be adapted to the particulars of your system (see below).

When you've finished setting up makefile.include, build VASP:

make all



Easy installation of scientific software:

conda install -c conda-forge <package-name>

Advantages:

- Over 900 materials science package
- Over 300 million downloads in total
- Distribution of precompiled packages
 - Linked to defined Python version
 - Dependency of packages clarified
 - Operating system independence

Conda-Forge

For Materials Science

matplotlib pandas ase numpy pytorch bokeh tensorflow dask nglview scikit-learn jupyter pycalphad phonopy before in Conda-Forge

Conda-Forge For Materials Science





Easy installation of scientific software:

conda install -c conda-forge <package-name>

Advantages:

- Over 900 materials science package
- Over 300 million downloads in total
- Distribution of precompiled packages
 - Linked to defined Python version
 - Dependency of packages clarified
 - Operating system independence



Conda-Forge For Materials Science







Conda-Forge For Materials Science





openkim	Easy installation of scientific software					
pyiron ovito atomskaflow	conda install	-c conda-forge <pacl< th=""><th>kage-name></th></pacl<>	kage-name>			
matplo pandaConda-forge bed open-source sc	came the default solut oftware in the field of r	ion for distributing naterials science.	ntages: ackage			
bytorch boken mendeleev tensorflow dask clease nglview sphinxdft scikit-learn jupyter atomicrex pycalphad phonopy	qsgenerator pymatgen	 Distribution of precomp Linked to defined F Dependency of pack 	iled packages Python version kages clarified			
before in Conda-Forge phonolammps		 Operating system 	independence			



pyiron – Complex Workflows Made Easy

pyiron – Complex Workflows Made Easy

```
Generic Interface
def rosen(x):
    """The Rosenbrock function"""
     return sum(
        100.0*(x[1:]-x[:-1]**2)**2 +
        (1-x[:-1])**2
 x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])
 res = minimize(
     rosen,
    x0,
    method="nelder-mead",
    options={"xatol": 1e-8, "disp": True}
 Optimization terminated successfully.
         Current function value: 0.000000
SciPy Iterations: 339
         Function evaluations: 571
```

pyiron – Complex Workflows Made Easy

Generic Interface def rosen(x): """The Rosenbrock function""" return sum(100.0*(x[1:]-x[:-1]**2)**2 +(1-x[:-1])**2x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2]) res = minimize(^{rosen}, 14 methods x0, method="nelder-mead" options={"xatol": 1e-8, "disp": True} Optimization terminated successfully. Current function value: 0.000000 SciPy Iterations: 339 Function evaluations: 571



pyiron – Complex Workflows Made Easy





Typical DFT calculation can take several minutes to hours. We need a technical abstraction for calculation and data management.





pyiron – Complex Workflows Made Easy





Typical DFT calculation can take several minutes to hours. We need a technical abstraction for calculation and data management.



Extendability

Scientific software is changing constantly and depending on your application you need a different set of tools.


Goals



pyiron – Complex Workflows Made Easy





: • •

Typical DFT calculation can take several minutes to hours. We need a technical abstraction for calculation and data management.



Extendability

Scientific software is changing constantly and depending on your application you need a different set of tools.



Goals



pyiron – Complex Workflows Made Easy





: • •

Typical DFT calculation can take several minutes to hours. We need a technical abstraction for calculation and data management.



Extendability

Scientific software is changing constantly and depending on your application you need a different set of tools.



Goals



pyiron – Complex Workflows Made Easy



Simulation Life Cycle



Workflow Provenance



Simulation Life Cycle

Workflow Provenance





Language Interface

Simulation Life Cycle



Workflow Provenance



Simulation Workflows



pyiron Enables Rapid Prototyping and Up-Scaling

```
In [1]: from pyiron import Project
```

```
In [2]: pr = Project("demonstration")
```

```
In [3]: job = pr.create.job.Lammps("my_calculation")
```

```
In [4]: job.structure = pr.create.structure.ase.bulk("Cu")
```

```
In [5]: # job.list_potentials()
```

In [6]: job.potential = "2012--Mendelev-M-I--Cu--LAMMPS--ipr1"

In [7]: job.server.cores = 2

In [8]: job.run()

The job my_calculation was saved and received the ID: 2

Simulation Workflows



pyiron Enables Rapid Prototyping and Up-Scaling

```
In [1]: from pyiron import Project
```

```
In [2]: pr = Project("demonstration")
```

```
In [3]: job = pr.create.job.Lammps("my_calculation")
```

In [4]: job.structure = pr.create.structure.ase.bulk("Cu")

```
In [5]: # job.list_potentials()
```

In [6]: job.potential = "2012--Mendelev-M-I--Cu--LAMMPS--ipr1"

In [7]: job.server.cores = 2

In [8]: job.run()

The job my_calculation was saved and received the ID: 2

Simulation Workflows



pyiron Enables Rapid Prototyping and Up-Scaling



The job my_calculation was saved and received the ID: 2

pyiron Objects Interfaces





pyiron Objects Interfaces









Collecting Results with the pyiron Table Object

```
[1]: from pyiron import Project # Import Project object
```

```
[2]: pr = Project("demonstration") # Create a project/folder
```

```
[3]: structure = pr.create.structure.bulk("Al") # Create an aluminium bulk structure
```

```
[4]: for job_type in ["Lammps", "Sphinx"]: # Iterate over simulation codes
    job = pr.create_job( # Create a job object
        job_type=job_type,
        job_name=job_type
    )
    job.structure = structure # assign the structure
    job.run() # Execute the calculation
```



Collecting Results with the pyiron Table Object

```
[1]: from pyiron import Project # Import Project object
[2]: pr = Project("demonstration") # Create a project/folder
[3]: structure = pr.create.structure.bulk("Al") # Create an aluminium bulk structure
[4]: for job_type in ["Lammps", "Sphinx"]: # Iterate over simulation codes
job = pr.create_job( # Create a job object
job_type=job_type,
job_name=job_type
)
job.structure = structure # assign the structure
job.run() # Execute the calculation
```



Collecting Results with the pyiron Table Object

```
[1]: from pyiron import Project # Import Project object
```

```
[2]: pr = Project("demonstration") # Create a project/folder
```

```
[3]: structure = pr.create.structure.bulk("Al") # Create an aluminium bulk structure
```

```
[4]: for job_type in ["Lammps", "Sphinx"]: # Iterate over simulation codes
    job = pr.create_job( # Create a job object
        job_type=job_type,
        job_name=job_type
    )
    job.structure = structure # assign the structure
    job.run() # Execute the calculation
```

[5]: table = pr.create_table() # Create analysis object table.add.get_energy_tot # Define analysis functions table.add.get_volume # get the volume and total energy table.add["job_type"] = lambda job: job.__name__ # Add custom analysis function table.run() # Execute the analysis table.get_dataframe() # Results are summarized in DataFrame



Collecting Results with the pyiron Table Object

```
[1]: from pyiron import Project # Import Project object
```

```
[2]: pr = Project("demonstration") # Create a project/folder
```

```
[3]: structure = pr.create.structure.bulk("Al") # Create an aluminium bulk structure
```

```
[4]: for job_type in ["Lammps", "Sphinx"]: # Iterate over simulation codes
        job = pr.create_job( # Create a job object
            job_type=job_type,
            job_name=job_type
                                                                job_id energy_tot volume job_type
        job.structure = structure # assign the structure
        job.run() # Execute the calculation
                                                                     1
                                                                          -3.36000 16.607531
                                                            0
                                                                                                   Lammps
[5]: table = pr.create table() # Create analysis object
                                                             1
                                                                     2
                                                                          -57.19579 16.607505
                                                                                                     Sphinx
    table.add.get energy tot # Define analysis functions
    table.add.get_volume # get the volume and total energy
    table.add["job type"] = lambda job: job. name
                                                   # Add custom analysis function
    table.run() # Execute the analysis
    table.get dataframe() # Results are summarized in DataFrame
```

Collecting Results with the pyiron Table Object

	m pyiron import Project # Import Project object								
	<pre>pr = Project("demonstration") # Create a project/folder</pre>	ect("demonstration") # Create a project/folder							
	<pre>structure = pr.create.structure.bulk("Al") # Create an a</pre>	ate.structure.bulk("Al") # Create an aluminium bulk structure							
[4]	Iterate over interatomic potentials or even simulation codes and afterwards collect the results using map-reduce.								
L		_				уре			
L	<pre>job.structure = structure # assign the structure job.run() # Execute the calculation</pre>	0	1	-3.36000	16.607531	Lammps			

Software for Materials Science

Automate Your Research - One Step at a Time





Maintaining over 900 materials informatics packages on conda-forge with over 300 million downloads so far. pyiron Workflow Framework



Jupyter Lab based interface for datadriven materials informatics to enable rapid prototyping and up-scaling.

.ar	ge l	Langu	LangS Jage N	im Iodel Ir	nterface		
+ %			Code 🗸	Notebook 🗋 🔅 I	Python 3 (ipykernel) $\bigcirc \equiv$		
[1]:	%reload_e	ext langsim					
[2]:	e of 200 GPa. EMT for me?						
	<pre>> Enterin Invoking: numbers=[2.04], [2 Invoking: s': [79], 0, 2.04]</pre>	Entering new AgentExecutor chain voking: `get_atom_dict_bulk_structure` with `{'chemical_symbol': 'Au'}` mbers=[79] positions=[[0.0, 0.0, 0.0]] cell=[[0.0, 2.04, 2.04], [2.04, 0.0, 04], [2.04, 2.04, 0.0]] pbc=[True, True, True] voking: `get_atom_dict_equilibrated_structure` with `{'atom_dict': {'number : [79], 'positions': [[0.0, 0.0, 0.0]], 'cell': [[0.0, 2.04, 2.04], [2.04, 0. 2.041, [2.04, 2.04, 0.01], 'pbc': [True, True]}. 'calculator str': 'em					
	t'}`	, [2.04, 2.04, 0	0.0]], pbc . [1	ue, ilue, iluej;, (
	St LBFGS: LBFGS:	tep Time 0 19:55:11 1 19:55:11	Energy 0.002606 0.000032	fmax 0.308859 0.077808			
	LBFGS: LBFGS: LBFGS:	2 19:55:11 3 19:55:11 4 19:55:11	-0.000135 -0.000135 -0.000135	0.003099 0.000029 0.000000			

Workflows developed for high throughput screening are now accessible through language interfaces.

Software for Materials Science

Automate Your Research - One Step at a Time







0

0 0

0

0 0

0

0 0

0 0

0

0

0 0 0 0 0

0

0 0



0 0 0 0 0 0 0 0 0 0 0



0 0 0 0 0 0 0 0 0 0 **Application of the pyiron Workflow Framework** 0 0 0 0 to ab-initio Thermodynamics 0 0 0 0 0 0

0 0

0 0





How to design ductile Magnesium alloys?

Pure Magnesium is brittle



10% Compression (CR)



How to design ductile Magnesium alloys?

Pure Magnesium is brittle



10% Compression (CR)





Z. Huang, et.al. Sci Rep 8, 3570 (2018)

How to design ductile Magnesium alloys?

Pure Magnesium is brittle



10% Compression (CR)

Dislocation Slip: Pure Magnesium





How to design ductile Magnesium alloys?

Pure Magnesium is brittle



10% Compression (CR)



Dislocation Slip: Pure Magnesium



Dislocation Slip: Magnesium Yttrium Alloy





How to design ductile Magnesium alloys?



Dislocation Slip: Pure Magnesium



Dislocation Slip: Magnesium Yttrium Alloy



How to design ductile Magnesium alloys?



How to design ductile Magnesium alloys?





Data-Driven Design Metal Alloys How to design ductile Magnesium alloys?





Materials Modelling



Hierarchical Nature of Structural Materials



Materials Modelling



Hierarchical Nature of Structural Materials



Materials Modelling

Hierarchical Nature of Structural Materials





Ab-initio Thermodynamics

Adiabatic Approach



$$F(V,T) = E^{tot}(V)$$

+ $F^{el}(V,T)$
+ $F^{qh}(V,T)$
+ $F^{ah}(V,T)$

Ab-initio Thermodynamics

Adiabatic Approach



$$F(V,T) = E^{tot}(V) +F^{el}(V,T) +F^{qh}(V,T) +F^{ah}(V,T)$$

Ab-initio Thermodynamics



Adiabatic Approach



Adiabatic Approach



$F(V,T) = E^{tot}(V) + F^{el}(V,T) + F^{qh}(V,T) + F^{ah}(V,T)$



Adiabatic Approach



$F(V,T) = E^{tot}(V) + F^{el}(V,T) + F^{qh}(V,T) + F^{ah}(V,T)$


Adiabatic Approach



$F(V,T) = E^{tot}(V) + F^{el}(V,T) + F^{qh}(V,T) + F^{ah}(V,T)$



Adiabatic Approach



$F(V,T) = E^{tot}(V) + F^{el}(V,T) + F^{qh}(V,T) + F^{ah}(V,T)$



















Divers Training Sets



Automated Small SYmmetric Structure Training (ASSYST)

Space Group Symmetry based Structures



Divers Training Sets

Automated Small SYmmetric Structure Training (ASSYST)



MPI for Sustainable Materials | Jan Janssen

Divers Training Sets



Automated Small SYmmetric Structure Training (ASSYST)



MPI for Sustainable Materials | Jan Janssen

M. Poul, L. Huber, E. Bitzek and J. Neugebauer., Phys. Rev. B 107 (2023)









For Spectral Neighbour Analysis Potential (SNAP)





on

Computational Cost calculating Molecular Dynamics





Computational Cost calculating Molecular Dynamics























Including Computational Cost



The cut-off radius is primarily a numerical hyperparameter, considerations based on the radial distribution function are less relevant.



Melting Temperature for Interatomic Force Fields





Melting Temperature for Interatomic Force Fields



Melting Temperature for Interatomic Force Fields



24



Melting Temperature for Interatomic Force Fields





Melting Temperature for Interatomic Force Fields



L.F. Zhu, J. Janssen, et al. Comp. Mat. Sci. 187 (2021)

Temperature Concentration Phase Diagram



Ab-initio Thermodynamics



MPI for Sustainable Materials | Jan Janssen

S. Menon, Y. Lysogorskiy, A. L. M. Knoll, N. Leimeroth, M. Poul, M. Qatar, J. Janssen, M. Mrovec, J. Rohrer, K. Albe, J. Behler, R. Drautz and J. Neugebauer. npj Comput. Mater., 10, 261 (2024)

Temperature Concentration Phase Diagram



Ab-initio Thermodynamics



MPI for Sustainable Materials | Jan Janssen

S. Menon, Y. Lysogorskiy, A. L. M. Knoll, N. Leimeroth, M. Poul, M. Qatar, J. Janssen, M. Mrovec, J. Rohrer, K. Albe, J. Behler, R. Drautz and J. Neugebauer. npj Comput. Mater., 10, 261 (2024)

25

Temperature Concentration Phase Diagram



Ab-initio Thermodynamics



J. Rohrer, K. Albe, J. Behler, R. Drautz and J. Neugebauer. npj Comput. Mater., 10, 261 (2024)

















Develop a Machine Learning Potential in Two Hours



Testing on Frontier



Develop a Machine Learning Potential in Two Hours



Testing on Frontier



Develop a Machine Learning Potential in Two Hours Testing on Frontier







Data-driven Atomistic Simulation



Impact of the pyiron workflow framework for ab-initio thermodynamics


Data-driven Atomistic Simulation



Impact of the pyiron workflow framework for ab-initio thermodynamics





0

0 0 0 0 0 0 0

0 0

0 0 0 0

0 0

0 0

0 0

0 0

0





0 0 0

0 0

0 0 0 0

0

Bridge the Gap Between Theory and Experiment

0	0	0	0
0	0	•	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0 0 0 **0 0**

0 0 0

Materials Acceleration Platform



Closed Loop High-throughput Screening for Battery Materials



Materials Acceleration Platform



Closed Loop High-throughput Screening for Battery Materials



Couple Experiments and Density Functional Theory





Chemical Gradient

MPI for Sustainable Materials | Jan Janssen

Couple Experiments and Density Functional Theory





Experiment

MPI for Sustainable Materials | Jan Janssen

Couple Experiments and Density Functional Theory







M. Stricker, L. Banko, N. Sarazin, N. Siemer, J. Janssen, L. Zhang, J. Neugebauer, A. Ludwig, arXiv 2212.04804 (2025). RUB

Couple Experiments and Density Functional Theory



M. Stricker, L. Banko, N. Sarazin, N. Siemer, J. Janssen, L. Zhang, J. Neugebauer, A. Ludwig, arXiv 2212.04804 (2025).

RUB

Couple Experiments and Density Functional Theory



M. Stricker, L. Banko, N. Sarazin, N. Siemer, J. Janssen, L. Zhang, J. Neugebauer, A. Ludwig, arXiv 2212.04804 (2025). RUB







MPI for Sustainable Materials | Jan Janssen

On the Materials Science Community

Open-source Development

≡	(0		Sign in		
Depriment pyiron (Put	olic	🗘 Notification	s 😵 Fork 50	☆ Star 383		
<> Code O Issues 4	î Pull requests 2	🖓 Discussion	s 🕑 Actions			
알 main ▾ 알 ♡ 좋 jan-janssen Merge	Go to file ✓ 1a185f2 · 3 w	<> Code	About NUMFEDCUS OPEN CODE = BETTER SCIENCE pyiron - an integrated development environment (IDE)			
ici_support	Update environ	3 weeks ago	science.	Inatenais		
devcontainer	Update devcont	5 years ago	🔗 pyiron.org			
.github	Remove conda d	6 months ago	python dft sir	mulation ide		
binder	Update environ	3 years ago	molecular-dynamics hdf5 lammps	vasp		

Hackathon in November 2024



On the Materials Science Community

Open-source Development

≡		()		Sign in
Depriment pyiron (Put	olic	D Notifications	양 Fork 50	☆ Star 383
<> Code 🕑 Issues 4	រ៉ា Pull requests	2 🖓 Discussions	s 🕑 Actions	•
알 main ▾ 양 ♡ 쥏 jan-janssen Merge	Go to file	<> Code •	About OPEN COL pyiron - an integ development en	FOCU DE = BETTER SCIEN rated vironment (ID
ci_support	Update environ	3 weeks ago	for computation science.	al materials
.devcontainer	Update devcont	5 years ago		
github	Remove conda d	6 months ago	python dft s	simulation ide
binder	Update environ	3 years ago	molecular-dynamics hdf5 lammps	s vasp



Hackathon in November 2024





THE UNIVERSITY OF BRITISH COLUMBIA

 (\bullet) INTERDISCIPLINARY CENTRE FOR







On the Materials Science Community

Open-source Development

≡		()		Sign in
Depriment pyiron (Pub	lic	Q Notifications	양 Fork 50	☆ Star 383
<> Code 🕑 Issues 4	រ៉ោ Pull requests	2 🖓 Discussion	s 🕑 Actions	•
१ main ▾ १ ▷ ② jan-janssen Merge	Go to file ✓ 1a185f2 · 3	<> Code →	About OPEN COL pyiron - an integr development env	F DUU E = BETTER SCIEN rated vironment (ID
ci_support	Update environ	3 weeks ago	for computationa science.	al materials
devcontainer	Update devcont	5 years ago	♂ pyiron.org	
.github	Remove conda d	6 months ago	python dft s	imulation ide
binder	Update environ	3 years ago	molecular-dynamics hdf5 lammps	vasp





Hackathon in November 2024





THE UNIVERSITY OF BRITISH COLUMBIA

NTERDISCIPLINARY CENTRE FOR







On the Materials Science Community

Open-source Development



Hackathon in November 2024



Growing number of external pyiron users, increasing number of open-source contributors and extension beyond atomistic simulation.



Workflows for data-driven Materials Science

Accelerate Your Research with the pyiron Workflow Framework





to atomistic simulation and prevents the hallucination of LLMs.

and outputs accelerate the parallel execution of Python workflows.

Materials Acceleration Platforms (MAP) combine simulation and experiment in the same workflow.

Workflows for data-driven Materials Science – Thank you



the same workflow.

Accelerate Your Research with the pyiron Workflow Framework



execution of Python workflows.

to atomistic simulation and prevents the hallucination of LLMs.