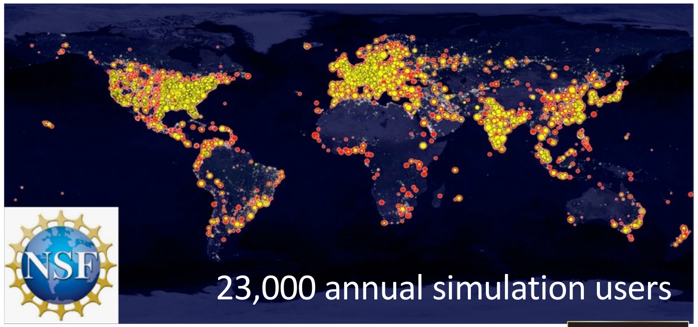


nanoHUB:
getting started guide to tool developers

Develop and publish tools in nanoHUB
Make your research reproducible and your workflows and data FAIR

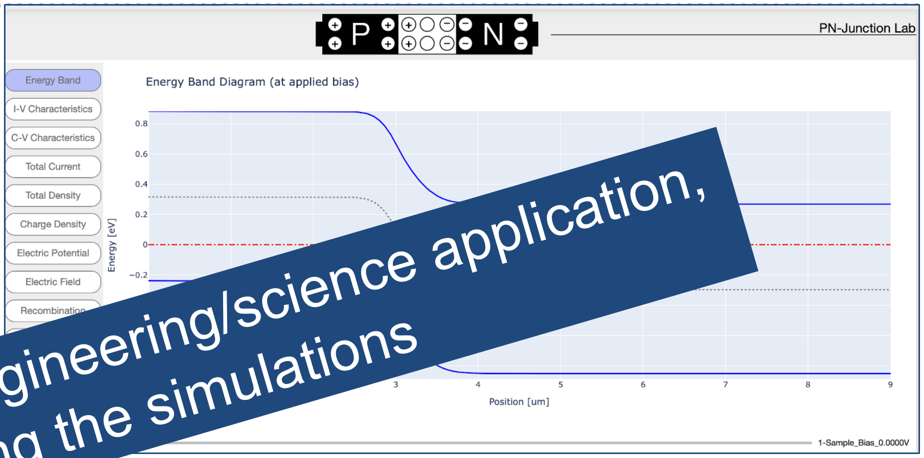
Tanya Faltens, Daniel Mejia, Steven Clark, Juan Carlos Verduzco & Ale Strachan*
* strachan@purdue.edu
School of Materials Engineering &
Purdue University
West Lafayette, Indiana USA

Overview

1. Why publish tools & apps in nanoHUB?
   - Tools are publications (DOIs and indexed by Web of Science)
   - Share your work with your community (22,000+ annual sim users)
2. Various tool and app types
   - Apps, workflows, Jupyter notebooks, commercial codes, X11 GUIs
3. Sim2Ls, FAIR workflows and data
   - Develop and publish Sim2Ls
4. Developing Apps
   - Connecting Sim2Ls to Jupyter and Web Apps
5. Tool Publication process
   - Register, deploy, test, and publish
6. Development environment
   - A Unix development environment (Jupyter or Linux desktop)
7. Simulation and data as a service
   - Launching tools and querying the ResultsDB

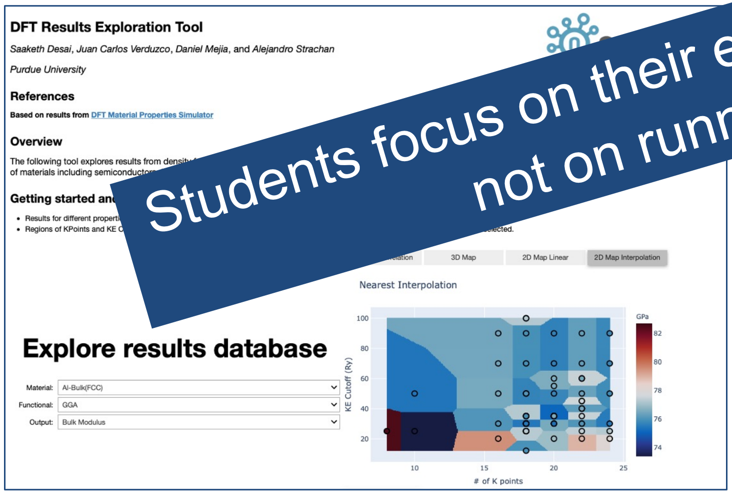
2. Tool types: Apps

Apps to focus on domain science & engineering, not on the computational aspects of the simulation



https://nanohub.org/tools/pnjunctionlab

Students focus on their engineering/science application, not on running the simulations



DFT Results Exploration Tool

Saaketh Desai, Juan Carlos Verduzco, Daniel Mejia, and Alejandro Strachan

Purdue University

References

Based on results from DFT Material Properties Simulator

Overview

The following tool explores results from density
of materials including semiconductors

Getting started and

- Results for different properti
- Regions of KPoints and KE C ... selected.

Explore results database

Material: Al-Bulk(FCC)

Functional: GGA

Output: Bulk Modulus

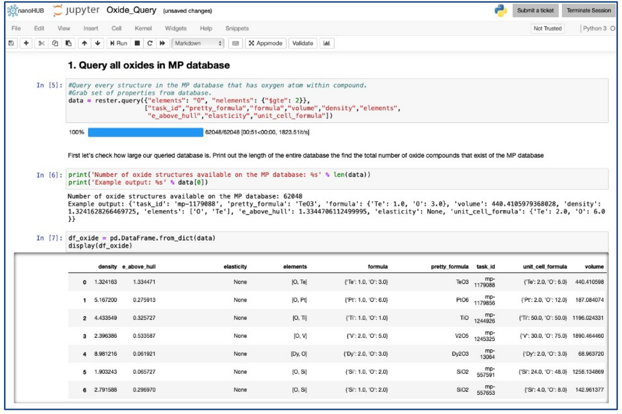
https://nanohub.org/tools/dftexplorer

Data exploration Apps

- Query the ResultsDB

Tool types: Jupyter notebooks

Data queries



1. Query all oxides in MP database

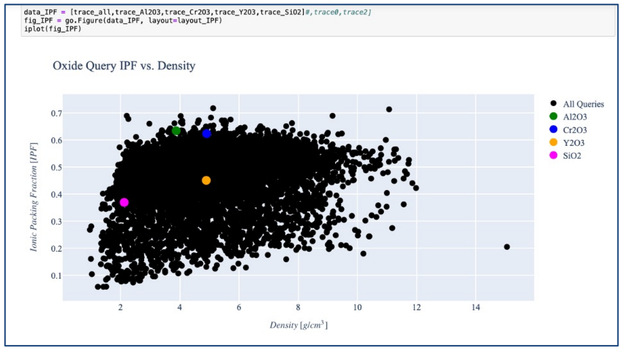
```
In [5]: #Query every structure in the MP database that has oxygen atom within compound.
        #Grab set of properties from database.
        data = rester.query({"elements": "O", "nelements": {"$gte": 2}},
                            ["task_id","pretty_formula","formula","volume","density","elements",
                             "e_above_hull","elasticity","unit_cell_formula"])

100% 62048/62048 [00:51<00:00, 1823.51it/s]
```

First let's check how large our queried database is. Print out the length of the entire database the find the total number of oxide compounds that exist of the MP database

```
In [6]: print('Number of oxide structures available on the MP database: %s' % len(data))
        print('Example output: %s' % data[0])

Number of oxide structures available on the MP database: 62048
Example output: {'task_id': 'mp-1179088', 'pretty_formula': 'TeO3', 'formula': {'Te': 1.0, 'O': 3.0}, 'volume': 440.4105979368028, 'density':
1.3241628266469725, 'elements': ['O', 'Te'], 'e_above_hull': 1.3344706112499995, 'elasticity': None, 'unit_cell_formula': {'Te': 2.0, 'O': 6.0
}}

In [7]: df_oxide = pd.DataFrame.from_dict(data)
        display(df_oxide)
```

	density	e_above_hull	elasticity	elements	formula	pretty_formula	task_id	unit_cell_formula	volume
0	1.324163	1.334471	None	[O, Te]	{'Te': 1.0, 'O': 3.0}	TeO3	mp-1179088	{'Te': 2.0, 'O': 6.0}	440.410598
1	5.167200	0.275913	None	[O, Pt]	{'Pt': 1.0, 'O': 6.0}	PtO6	mp-1179856	{'Pt': 2.0, 'O': 12.0}	187.084074
2	4.433549	0.325727	None	[O, Ti]	{'Ti': 1.0, 'O': 1.0}	TiO	mp-1244926	{'Ti': 50.0, 'O': 50.0}	1196.024331
3	2.396386	0.533587	None	[O, V]	{'V': 2.0, 'O': 5.0}	V2O5	mp-1245325	{'V': 30.0, 'O': 75.0}	1890.464460
4	8.981216	0.061921	None	[Dy, O]	{'Dy': 2.0, 'O': 3.0}	Dy2O3	mp-13064	{'Dy': 2.0, 'O': 3.0}	68.963720
5	1.903243	0.065727	None	[O, Si]	{'Si': 1.0, 'O': 2.0}	SiO2	mp-557591	{'Si': 24.0, 'O': 48.0}	1258.134869
6	2.791588	0.296970	None	[O, Si]	{'Si': 1.0, 'O': 2.0}	SiO2	mp-557653	{'Si': 4.0, 'O': 8.0}	142.961377

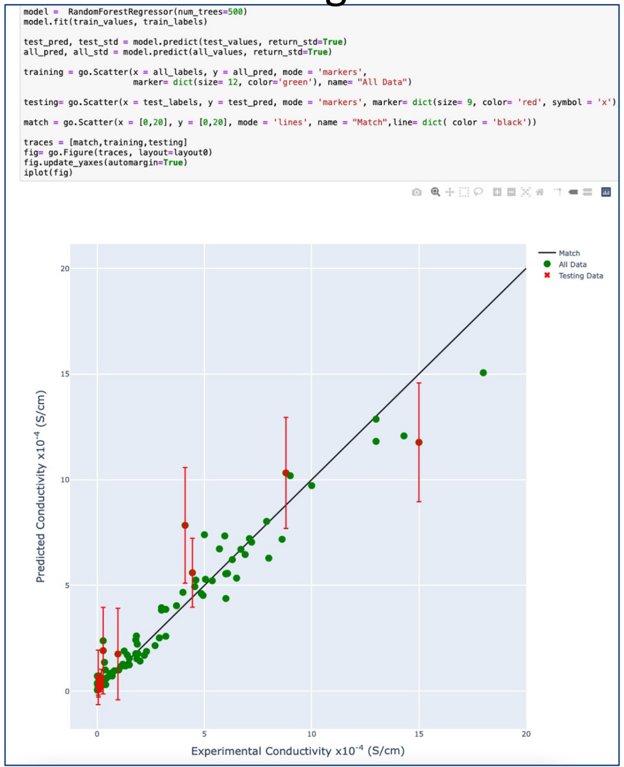
Visualization



```
data_IPF = [trace_all,trace_Al2O3,trace_Cr2O3,trace_Y2O3,trace_SiO2]#,trace0,trace2]
fig_IPF = go.Figure(data_IPF, layout=layout_IPF)
iplot(fig_IPF)
```

https://nanohub.org/tools/matdatarepo

Machine learning



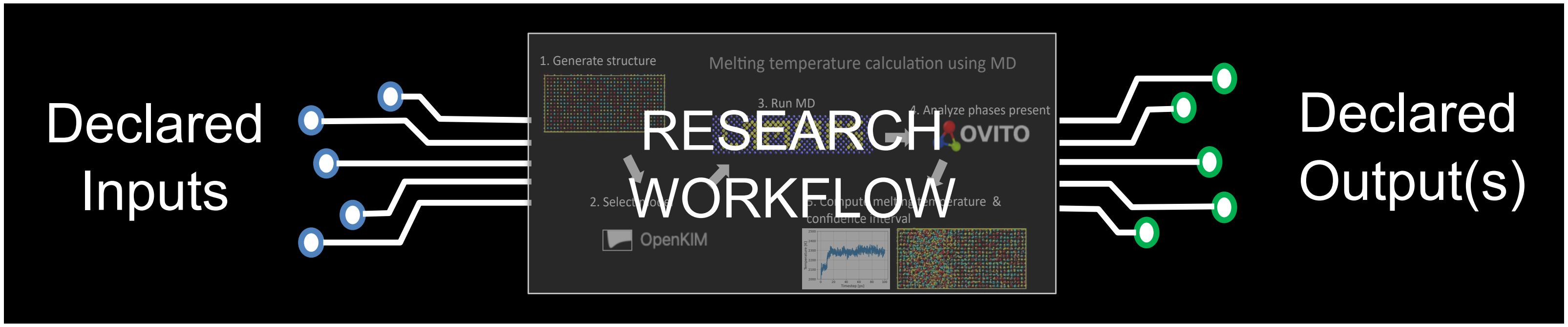
```
model =  RandomForestRegressor(num_trees=500)
model.fit(train_values, train_labels)

test_pred, test_std = model.predict(test_values, return_std=True)
all_pred, all_std = model.predict(all_values, return_std=True)

training = go.Scatter(x = all_labels, y = all_pred, mode = 'markers',
                      marker= dict(size= 12, color='green'), name= "All Data")

testing= go.Scatter(x = test_labels, y = test_pred, mode = 'markers', marker= dict(size= 9, color= 'red', symbol = 'x')

match = go.Scatter(x = [0,20], y = [0,20], mode = 'lines', name = "Match",line= dict( color = 'black'))

traces = [match,training,testing]
fig= go.Figure(traces, layout=layout0)
fig.update_yaxes(automargin=True)
iplot(fig)
```

Predicted Conductivity x10^{-4} (S/cm)
0
5
10
15
20
0
5
10
15
20
Experimental Conductivity x10^{-4} (S/cm)
Match
● All Data
✖ Testing Data

https://nanohub.org/tools/citrinetools

Tool types: Sim2Ls

- Full end-to-end computational workflow
  - Input(s) → workflow → output(s)
  - Including all pre-processing and post-processing steps

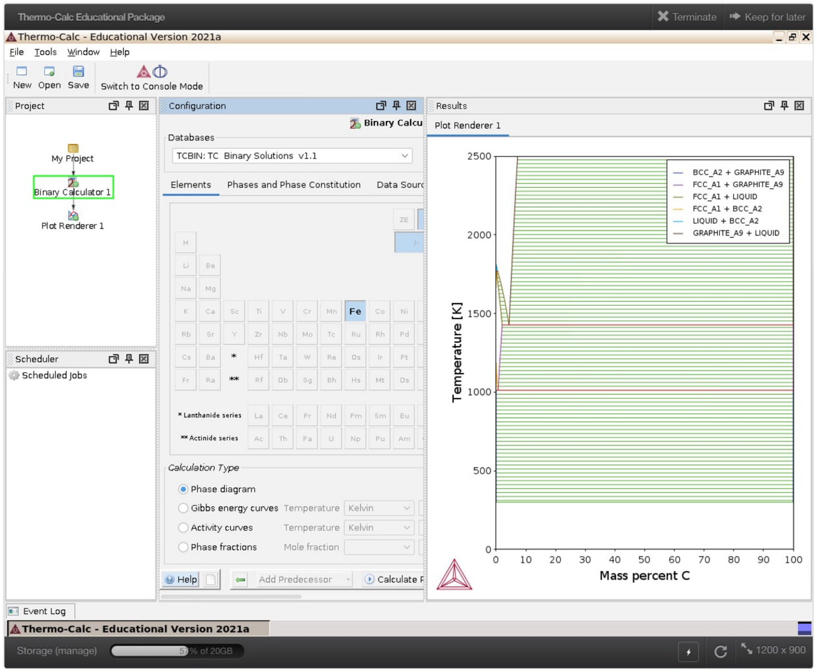

Declared Inputs
1. Generate structure
Melting temperature calculation using MD
3. Run MD
4. Analyze phases present
OVITO
RESEARCH WORKFLOW
2. Select model
OpenKIM
5. Compute melting temperature & confidence interval
Declared Output(s)

- Simulation as a service. Launch Sim2Ls from:
  - From a GUI or App
  - From AI/ML or high-throughput workflows
  - From inside nanoHUB or outside

Hunt M, Clark S, Mejia D, Desai S, Strachan A (2022) Sim2Ls: FAIR simulation workflows and data. PLoS ONE 17(3): e0264492.
https://doi.org/10.1371/journal.pone.0264492

Tool types: commercial codes

Thermo-Calc



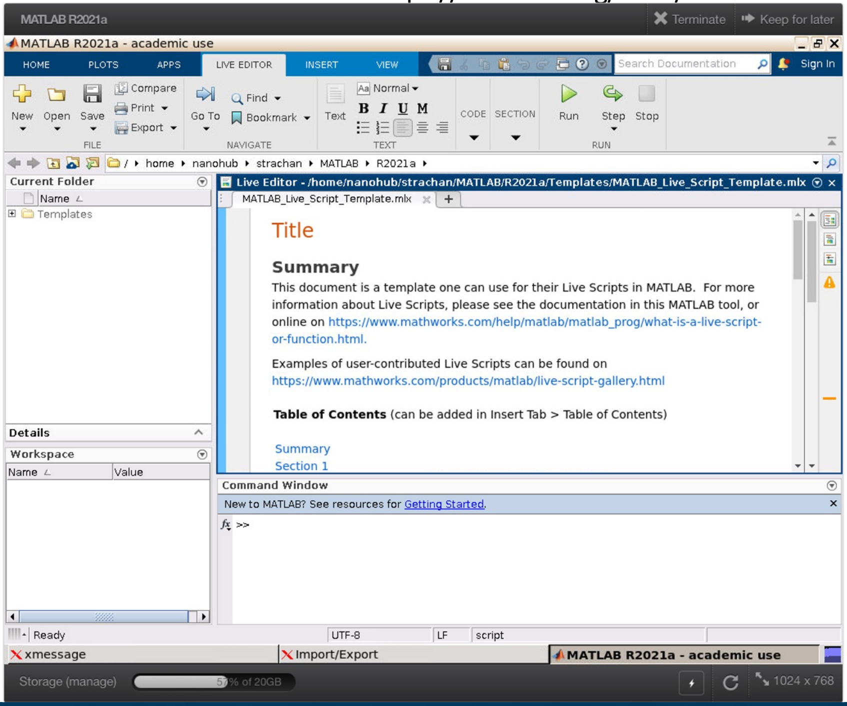
Databases: TCBIN: TC Binary Solutions v1.1

Calculation Type: ● Phase diagram

https://nanohub.org/tools/tcacademic

MATLAB

https://nanohub.org/tools/matlab2021a



Title

Summary

This document is a template one can use for their Live Scripts in MATLAB. For more information about Live Scripts, please see the documentation in this MATLAB tool, or online on https://www.mathworks.com/help/matlab/matlab_prog/what-is-a-live-script-or-function.html.

Examples of user-contributed Live Scripts can be found on https://www.mathworks.com/products/matlab/live-script-gallery.html

Table of Contents (can be added in Insert Tab > Table of Contents)

Summary
Section 1

New to MATLAB? See resources for Getting Started.

```
fx >>
```