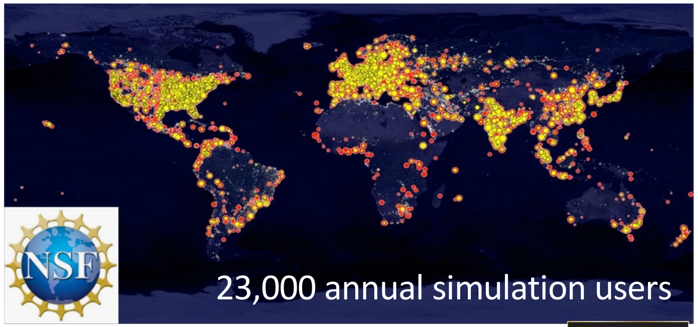


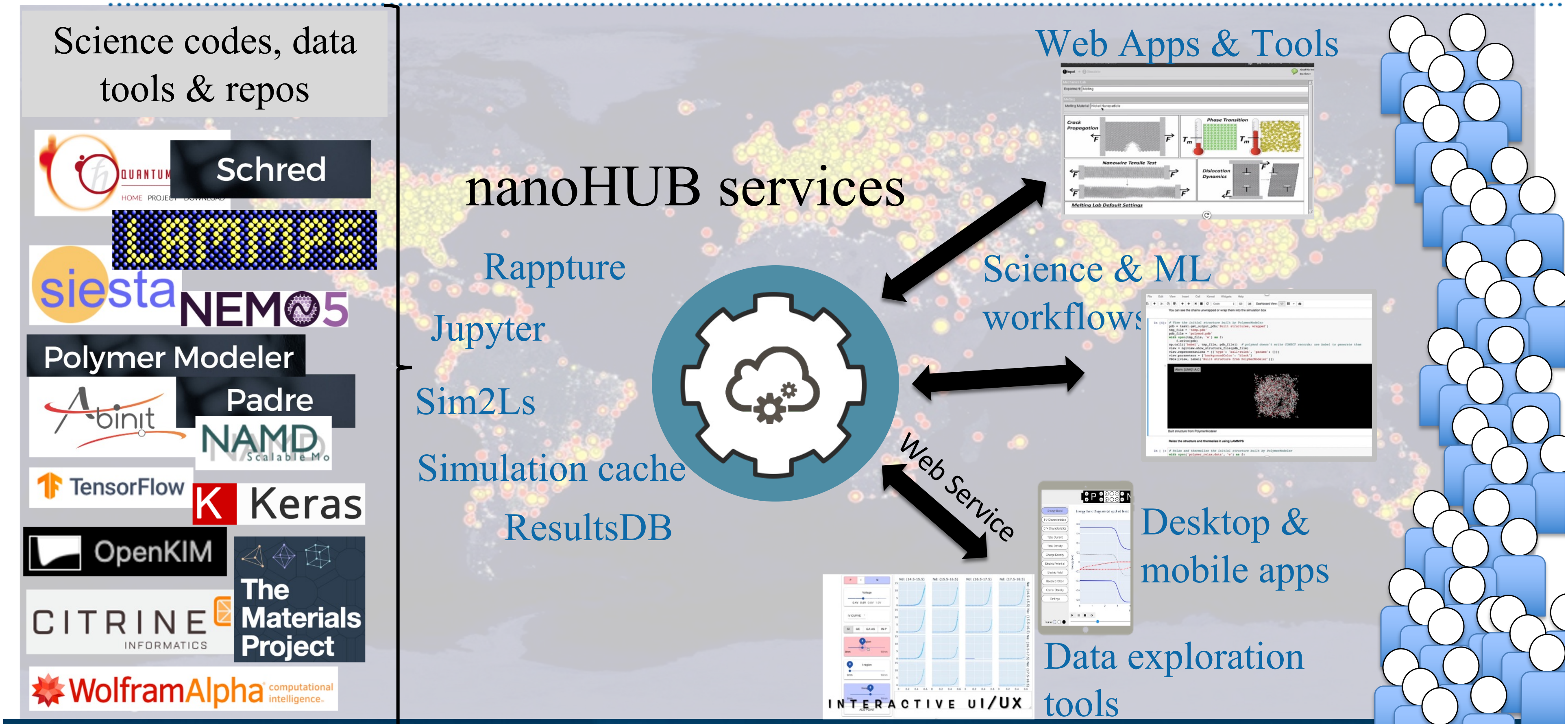
23,000 annual simulation users

nanoHUB's Sim2Ls
getting started guide for tool developers

Make your research reproducible and your workflows and data FAIR

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

nanoHUB: online apps, tools, & data



Science codes, data tools & repos

nanoHUB services

Rappture

Jupyter

Sim2Ls

Simulation cache

ResultsDB

Web Apps & Tools

Science & ML workflows

Web Service

Desktop & mobile apps

Data exploration tools

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

Why Sim2Ls?

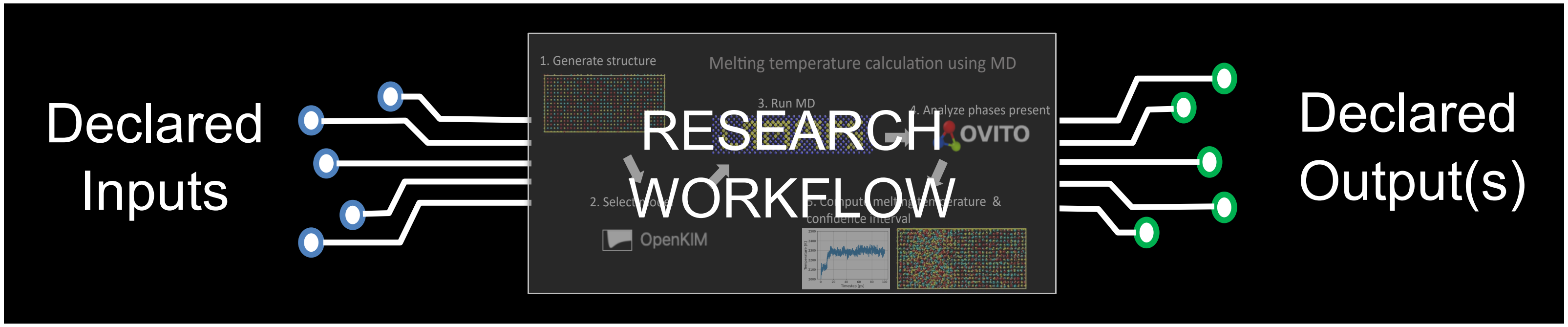
- Simulation and data analysis workflows :
  - Are complex and multi-step
  - (often) involve ad-hoc or manual steps
  - (often) only partially described in publications

- Consequently:
  - Reproducing results requires significant effort, even by experts
  - Domain experts (not computational experts) cannot benefit from these workflows
  - Scientific progress and innovation are hindered

- Workflows and the results they generate are not
  - Findable, accessible, interoperable, reusable (FAIR)
  - Reproducible

What are Sim2Ls?

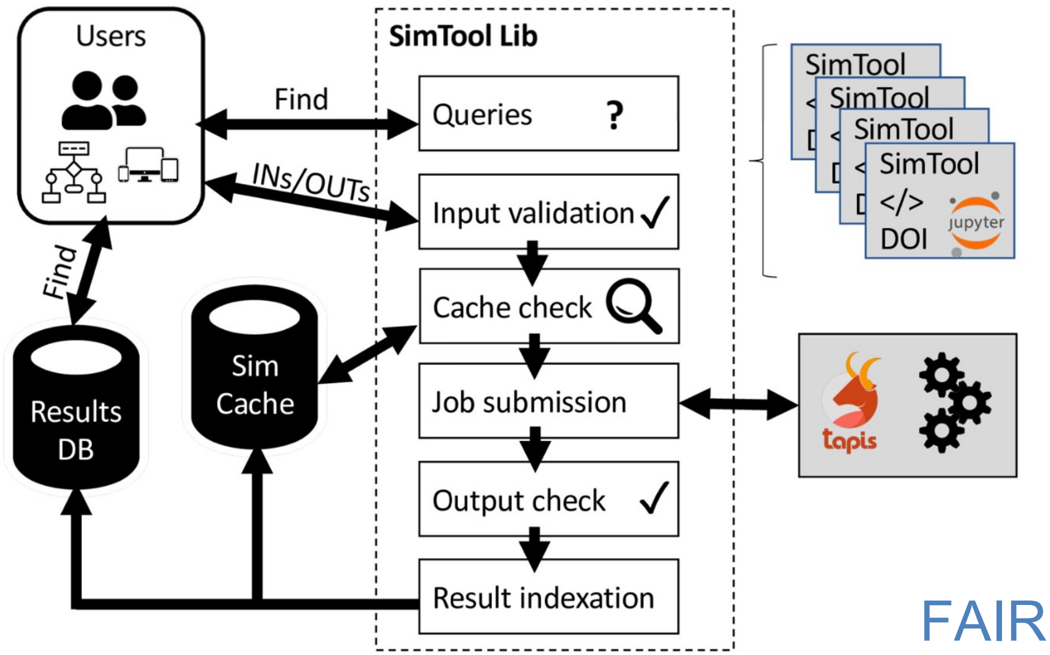
- Full end-to-end computational workflow
  - Input(s) → workflow → output(s)
  - Including all pre-processing and post-processing steps



- 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

Sim2Ls: key features



Users

Find

INs/OUTs

Find

Results DB

Sim Cache

SimTool Lib

Queries ?

Input validation ✓

Cache check

Job submission

Output check ✓

Result indexation

SimTool

SimTool

SimTool

SimTool

</>

DOI

jupyter

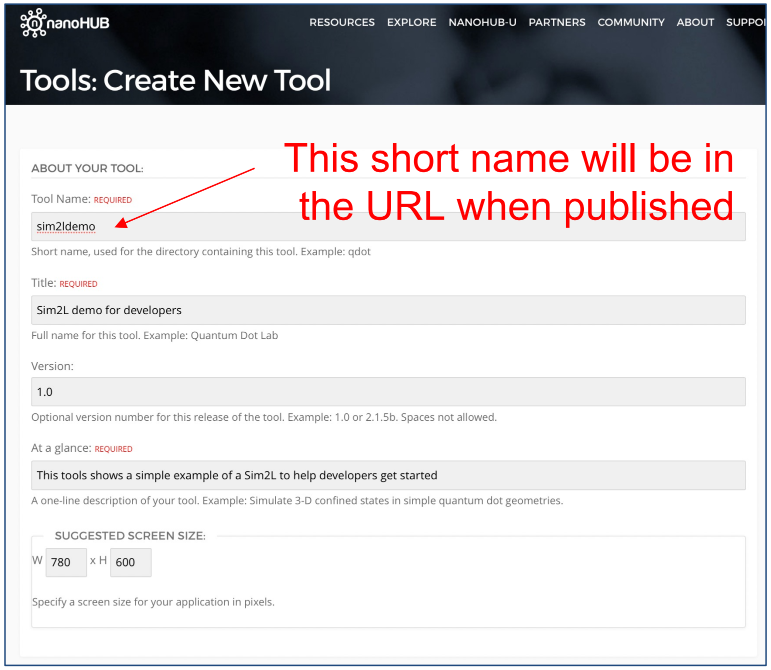
tapis

- Published Sim2Ls have DOIs and are indexed by Web Of Science
- Services (outputs) & requirements (inputs) are queryable
- Simulation are automatically stored in a cache and not re-run
- Simulation results are indexed in queryable database (ResultsDB)

FAIR workflows and data

Developing a Sim2L – Step 1: register your tool

Step 1: register your Sim2L:
https://nanohub.org/tools/create



Tools: Create New Tool

ABOUT YOUR TOOL:

Tool Name: REQUIRED
sim2ldemo
Short name, used for the directory containing this tool. Example: qdot

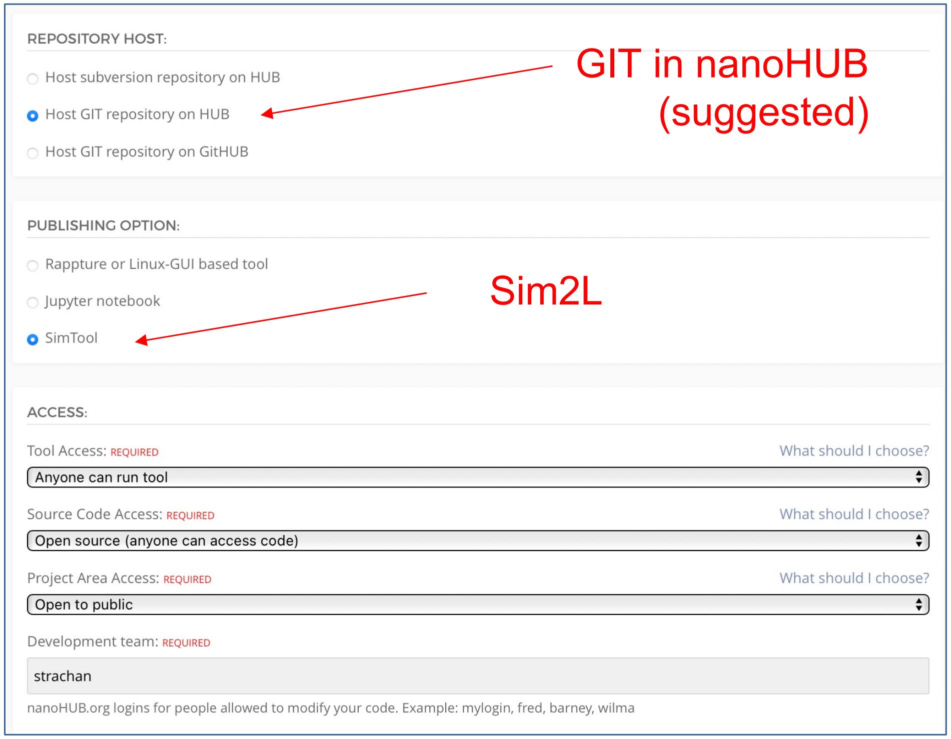
This short name will be in the URL when published

Title: REQUIRED
Sim2L demo for developers
Full name for this tool. Example: Quantum Dot Lab

Version:
1.0
Optional version number for this release of the tool. Example: 1.0 or 2.1.5b. Spaces not allowed.

At a glance: REQUIRED
This tools shows a simple example of a Sim2L to help developers get started
A one-line description of your tool. Example: Simulate 3-D confined states in simple quantum dot geometries.

SUGGESTED SCREEN SIZE:
W 780 x H 600
Specify a screen size for your application in pixels.



REPOSITORY HOST:

○ Host subversion repository on HUB
● Host GIT repository on HUB
○ Host GIT repository on GitHUB

GIT in nanoHUB (suggested)

PUBLISHING OPTION:

○ Rappture or Linux-GUI based tool
○ Jupyter notebook
● SimTool

Sim2L

ACCESS:

Tool Access: REQUIRED — What should I choose?
Anyone can run tool

Source Code Access: REQUIRED — What should I choose?
Open source (anyone can access code)

Project Area Access: REQUIRED — What should I choose?
Open to public

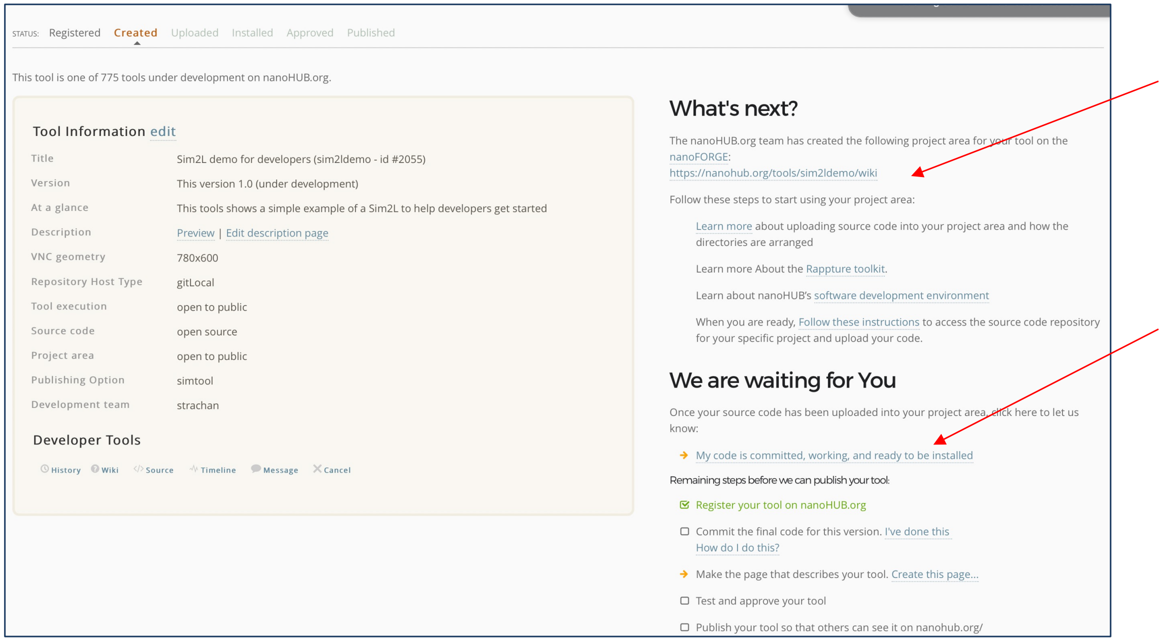
Development team: REQUIRED
strachan
nanoHUB.org logins for people allowed to modify your code. Example: mylogin, fred, barney, wilma

Click Register

Developing a Sim2L – Step 1: register your tool

Your tool is registered, you can start working on it

https://nanohub.org/tools/sim2ldemo/status



STATUS: Registered **Created** Uploaded Installed Approved Published

This tool is one of 775 tools under development on nanoHUB.org.

Tool Information edit

Title	Sim2L demo for developers (sim2ldemo - id #2055)
Version	This version 1.0 (under development)
At a glance	This tools shows a simple example of a Sim2L to help developers get started
Description	Preview | Edit description page
VNC geometry	780x600
Repository Host Type	gitLocal
Tool execution	open to public
Source code	open source
Project area	open to public
Publishing Option	simtool
Development team	strachan

Developer Tools

What's next?

The nanoHUB.org team has created the following project area for your tool on the nanoFORGE:
https://nanohub.org/tools/sim2ldemo/wiki

Follow these steps to start using your project area:

Learn more about uploading source code into your project area and how the directories are arranged

Learn more About the Rappture toolkit.

Learn about nanoHUB's software development environment

When you are ready, Follow these instructions to access the source code repository for your specific project and upload your code.

We are waiting for You

Once your source code has been uploaded into your project area, click here to let us know:

→ My code is committed, working, and ready to be installed

Remaining steps before we can publish your tool:

☑ Register your tool on nanoHUB.org

□ Commit the final code for this version. I've done this
How do I do this?

→ Make the page that describes your tool. Create this page...

□ Test and approve your tool

□ Publish your tool so that others can see it on nanohub.org/

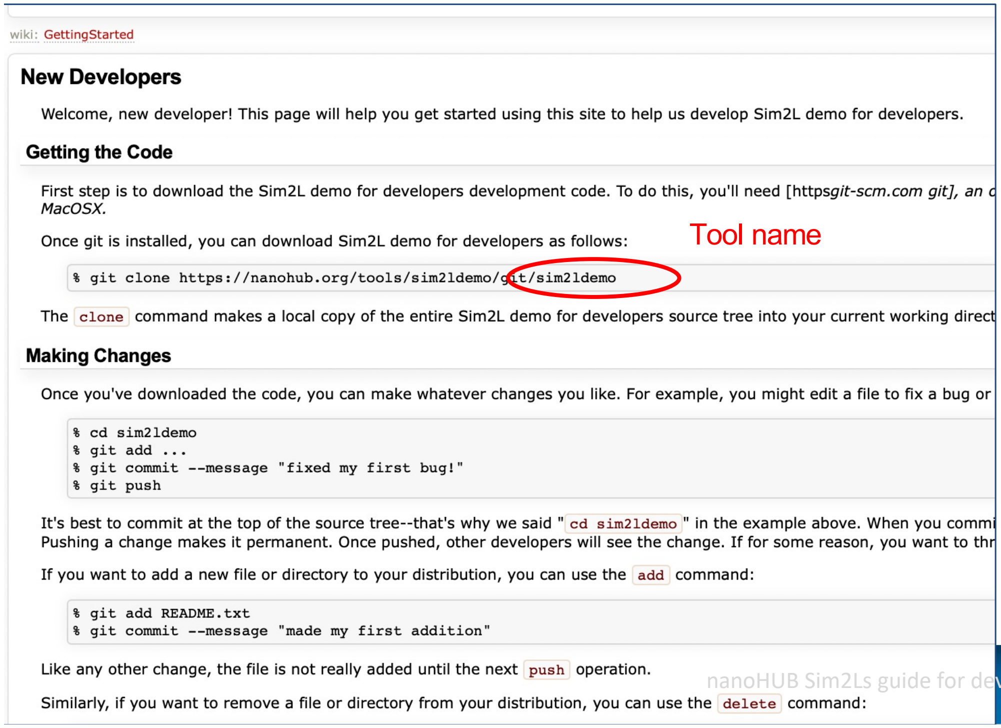
Go to the project area and start working

When you are done and committed your ready-to-test code click here (you will be able to further test it before it is deployed to the public)

Developing a Sim2L – Step 2: clone your tool’s repo

Step 2: clone the git repo in your nanoHUB workspace
Instructions at: https://nanohub.org/tools/*sim2demo*/wiki (click on Getting Started)

Your tool name



wiki: GettingStarted

New Developers

Welcome, new developer! This page will help you get started using this site to help us develop Sim2L demo for developers.

Getting the Code

First step is to download the Sim2L demo for developers development code. To do this, you'll need [httpsgit-scm.com git], an c *MacOSX.*

Once git is installed, you can download Sim2L demo for developers as follows:

```
% git clone https://nanohub.org/tools/sim2ldemo/git/sim2ldemo
```

The `clone` command makes a local copy of the entire Sim2L demo for developers source tree into your current working direct

Making Changes

Once you've downloaded the code, you can make whatever changes you like. For example, you might edit a file to fix a bug or

```
% cd sim2ldemo
% git add ...
% git commit --message "fixed my first bug!"
% git push
```

It's best to commit at the top of the source tree--that's why we said "`cd sim2ldemo`" in the example above. When you commi Pushing a change makes it permanent. Once pushed, other developers will see the change. If for some reason, you want to thr

If you want to add a new file or directory to your distribution, you can use the `add` command:

```
% git add README.txt
% git commit --message "made my first addition"
```

Like any other change, the file is not really added until the next `push` operation.

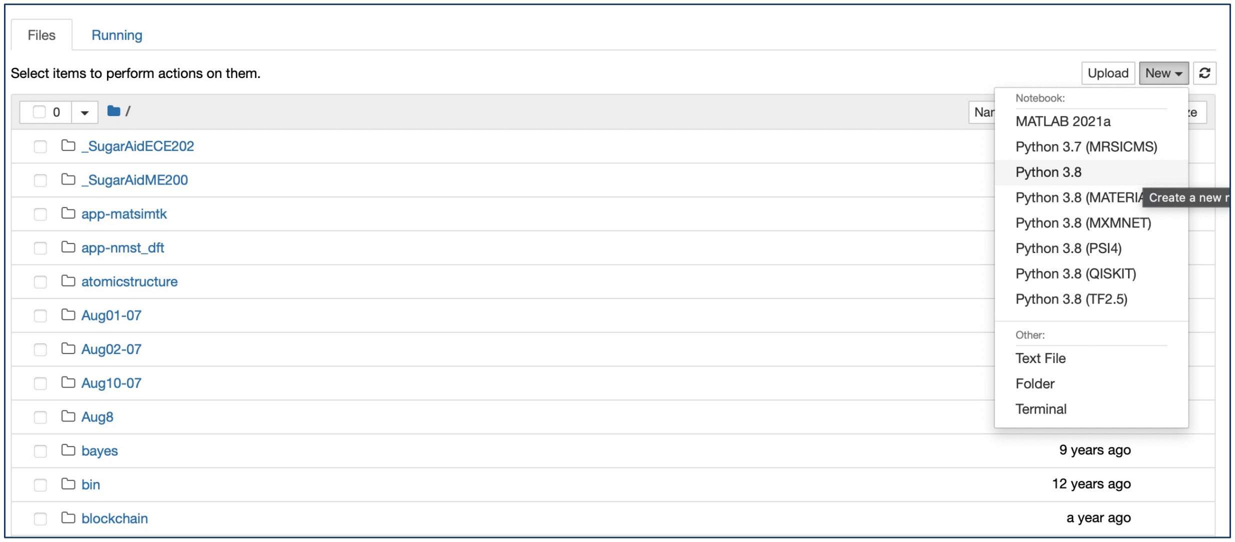
Similarly, if you want to remove a file or directory from your distribution, you can use the `delete` command:

Clone the automatically created GIT repo

Follow the instructions to make changes and commit them using GIT

Developing a Sim2L – Step 2: clone your tool's repo

Step 2: clone the git repo in your nanoHUB workspace
2.1 Launch Jupyter in nanoHUB: https://nanohub.org/tools/jupyter70

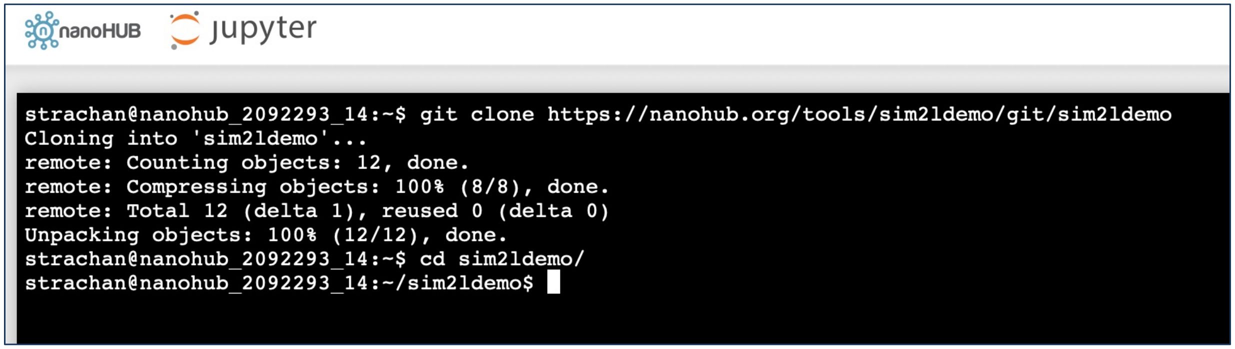


Select items to perform actions on them.

_SugarAidECE202	
_SugarAidME200	
app-matsimtk	
app-nmst_dft	
atomicstructure	
Aug01-07	
Aug02-07	
Aug10-07	
Aug8	
bayes	9 years ago
bin	12 years ago
blockchain	a year ago

Start a terminal and/or a python kernel

Developing a Sim2L – Step 3: work on your tool

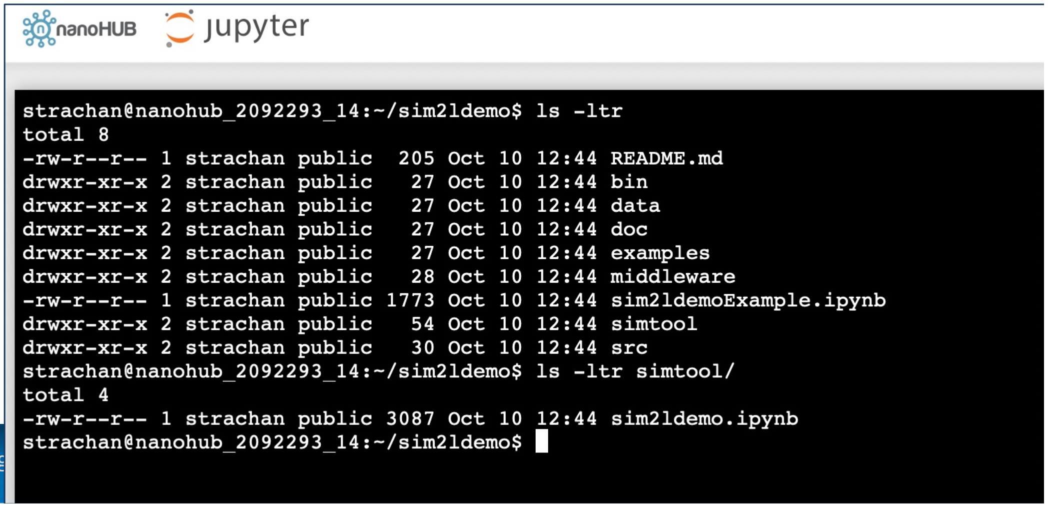


```
strachan@nanohub_2092293_14:~$ git clone https://nanohub.org/tools/sim2ldemo/git/sim2ldemo
Cloning into 'sim2ldemo'...
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 12 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (12/12), done.
strachan@nanohub_2092293_14:~$ cd sim2ldemo/
strachan@nanohub_2092293_14:~/sim2ldemo$ █
```

Clone and check out the directory structure and template files

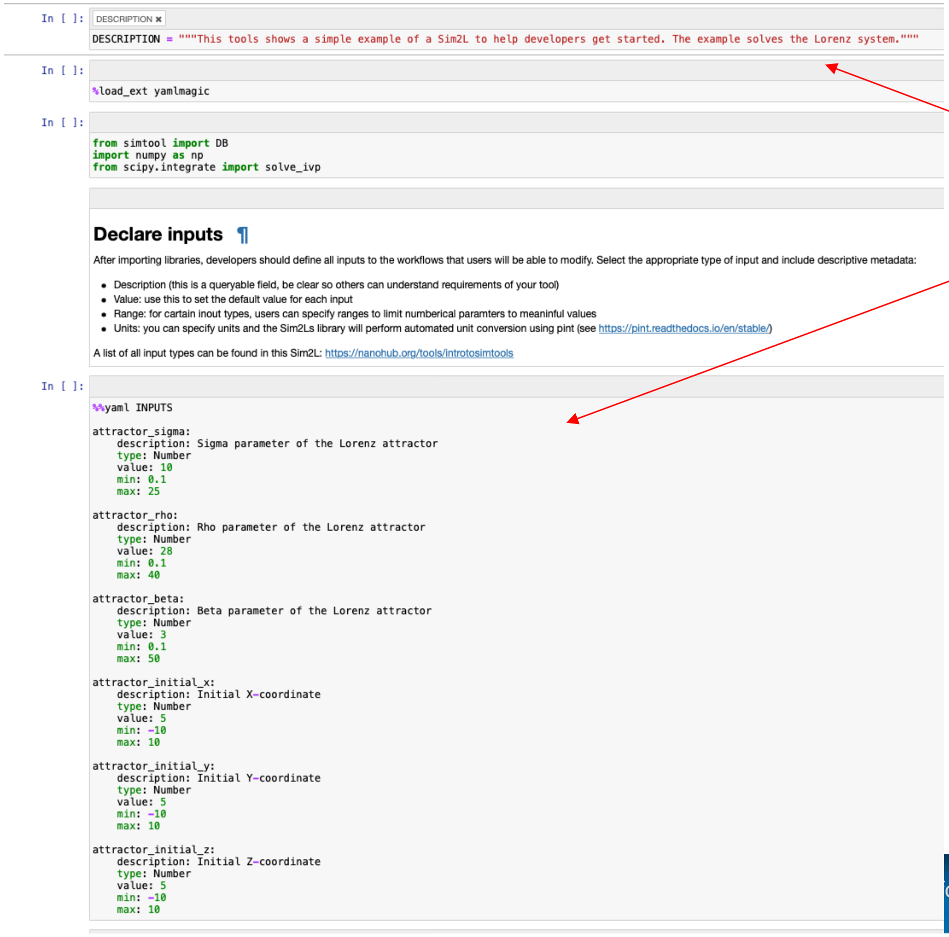
Two key files are created:

- `simtool/sim2ldemo.ipynb` (actual Sim2L workflow)
- `sim2ldemoExample.ipynb` (execution notebook to invoke the Sim2L)



```
strachan@nanohub_2092293_14:~/sim2ldemo$ ls -ltr
total 8
-rw-r--r-- 1 strachan public  205 Oct 10 12:44 README.md
drwxr-xr-x 2 strachan public   27 Oct 10 12:44 bin
drwxr-xr-x 2 strachan public   27 Oct 10 12:44 data
drwxr-xr-x 2 strachan public   27 Oct 10 12:44 doc
drwxr-xr-x 2 strachan public   27 Oct 10 12:44 examples
drwxr-xr-x 2 strachan public   28 Oct 10 12:44 middleware
-rw-r--r-- 1 strachan public 1773 Oct 10 12:44 sim2ldemoExample.ipynb
drwxr-xr-x 2 strachan public   54 Oct 10 12:44 simtool
drwxr-xr-x 2 strachan public   30 Oct 10 12:44 src
strachan@nanohub_2092293_14:~/sim2ldemo$ ls -ltr simtool/
total 4
-rw-r--r-- 1 strachan public 3087 Oct 10 12:44 sim2ldemo.ipynb
strachan@nanohub_2092293_14:~/sim2ldemo$ █
```

Developing a Sim2L – Step 3: Sim2L notebook (part 1)



```
In [ ]: DESCRIPTION = """This tools shows a simple example of a Sim2L to help developers get started. The example solves the Lorenz system."""
```

```
In [ ]: %load_ext yamlmagic
```

```
In [ ]: from simtool import DB
        import numpy as np
        from scipy.integrate import solve_ivp
```

Declare inputs ¶

After importing libraries, developers should define all inputs to the workflows that users will be able to modify. Select the appropriate type of input and include descriptive metadata:

- Description (this is a queryable field, be clear so others can understand requirements of your tool)
- Value: use this to set the default value for each input
- Range: for cartain inout types, users can specify ranges to limit numberical paramters to meaninful values
- Units: you can specify units and the Sim2Ls library will perform automated unit conversion using pint (see https://pint.readthedocs.io/en/stable/)

A list of all input types can be found in this Sim2L: https://nanohub.org/tools/introtosimtools

```
In [ ]: %%yaml INPUTS

attractor_sigma:
    description: Sigma parameter of the Lorenz attractor
    type: Number
    value: 10
    min: 0.1
    max: 25

attractor_rho:
    description: Rho parameter of the Lorenz attractor
    type: Number
    value: 28
    min: 0.1
    max: 40

attractor_beta:
    description: Beta parameter of the Lorenz attractor
    type: Number
    value: 3
    min: 0.1
    max: 50

attractor_initial_x:
    description: Initial X-coordinate
    type: Number
    value: 5
    min: -10
    max: 10

attractor_initial_y:
    description: Initial Y-coordinate
    type: Number
    value: 5
    min: -10
    max: 10

attractor_initial_z:
    description: Initial Z-coordinate
    type: Number
    value: 5
    min: -10
    max: 10
```

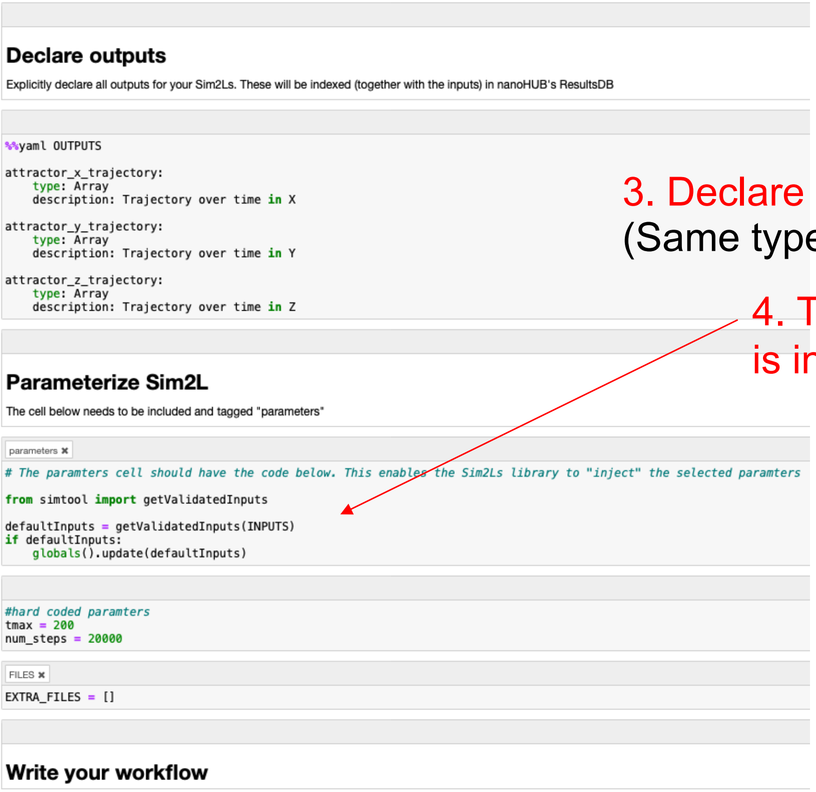
Sim2L template in the `simtool/` folder

1. A description is required and queryable

2. Declare all inputs (include descriptions and units if applicable)
(Possible input types: `Boolean, Integer, Number, Array, Text, Choice, List, Dictionary, Image, and Element)`

Check out https://nanohub.org/tools/introtosimtools for examples of all input types

Developing a Sim2L – Step 3: Sim2L notebook (part 2)



Declare outputs

Explicitly declare all outputs for your Sim2Ls. These will be indexed (together with the inputs) in nanoHUB's ResultsDB

```
%%yaml OUTPUTS

attractor_x_trajectory:
    type: Array
    description: Trajectory over time in X

attractor_y_trajectory:
    type: Array
    description: Trajectory over time in Y

attractor_z_trajectory:
    type: Array
    description: Trajectory over time in Z
```

Parameterize Sim2L

The cell below needs to be included and tagged "parameters"

parameters ✖

```
# The paramters cell should have the code below. This enables the Sim2Ls library to "inject" the selected paramters

from simtool import getValidatedInputs

defaultInputs = getValidatedInputs(INPUTS)
if defaultInputs:
    globals().update(defaultInputs)
```

```
#hard coded paramters
tmax = 200
num_steps = 20000
```

FILES ✖

```
EXTRA_FILES = []
```

Write your workflow

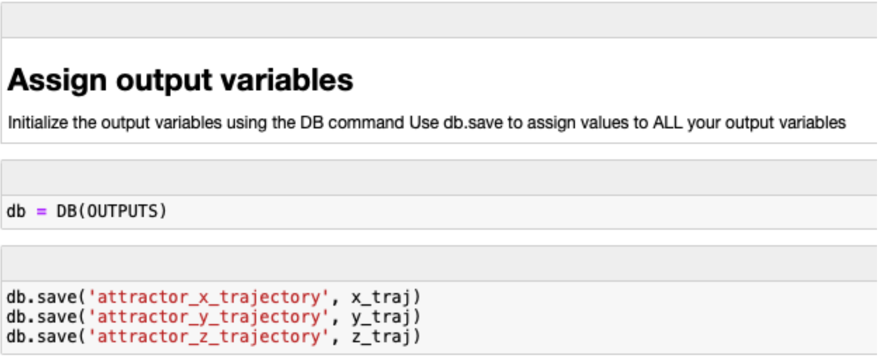
…

3. Declare all outputs (include descriptions)
(Same types as inputs)

4. This cell is used to inject the parameters when the Sim2L is invoked (leave it as is)

5. Write your workflow connecting inputs to outputs

6. Assign output variables by creating a database object



Assign output variables

Initialize the output variables using the DB command Use db.save to assign values to ALL your output variables

```
db = DB(OUTPUTS)
```

```
db.save('attractor_x_trajectory', x_traj)
db.save('attractor_y_trajectory', y_traj)
db.save('attractor_z_trajectory', z_traj)
```

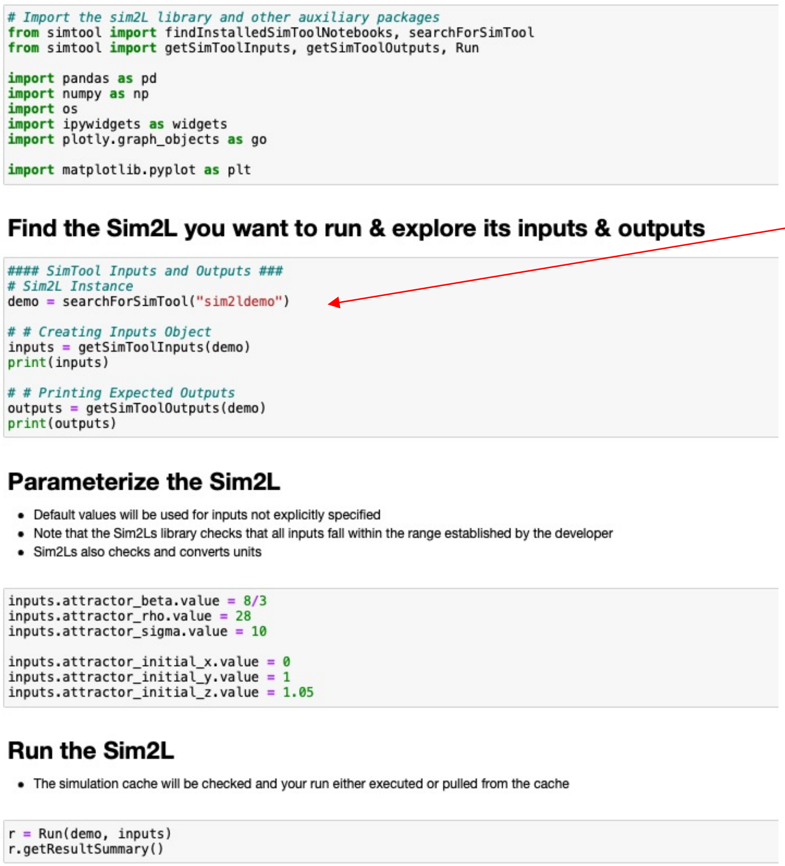
Developing a Sim2L – Step 3: execution notebook (Part 1)

This is the file that will be launched when the tool is run in nanoHUB
This notebook should set inputs, invoke the Sim2L itself, display outputs

Several options:

- A plain notebook that sets inputs and displays outputs
- A GUI (using widgets), see https://ipywidgets.readthedocs.io/en/stable/
- An AI/ML or high throughput workflow that launches the Sim2L as needed

Developing a Sim2L – Step 3: execution notebook (Part 2)



```
# Import the sim2L library and other auxiliary packages
from simtool import findInstalledSimToolNotebooks, searchForSimTool
from simtool import getSimToolInputs, getSimToolOutputs, Run

import pandas as pd
import numpy as np
import os
import ipywidgets as widgets
import plotly.graph_objects as go

import matplotlib.pyplot as plt
```

Find the Sim2L you want to run & explore its inputs & outputs

```
#### SimTool Inputs and Outputs ###
# Sim2L Instance
demo = searchForSimTool("sim2ldemo")

# # Creating Inputs Object
inputs = getSimToolInputs(demo)
print(inputs)

# # Printing Expected Outputs
outputs = getSimToolOutputs(demo)
print(outputs)
```

Parameterize the Sim2L

- Default values will be used for inputs not explicitly specified
- Note that the Sim2Ls library checks that all inputs fall within the range established by the developer
- Sim2Ls also checks and converts units

```
inputs.attractor_beta.value = 8/3
inputs.attractor_rho.value = 28
inputs.attractor_sigma.value = 10

inputs.attractor_initial_x.value = 0
inputs.attractor_initial_y.value = 1
inputs.attractor_initial_z.value = 1.05
```

Run the Sim2L

- The simulation cache will be checked and your run either executed or pulled from the cache

```
r = Run(demo, inputs)
r.getResultSummary()
```

1. Identify the Sim2L you want to execute

The tool can be published or under development

2. Check inputs and outputs

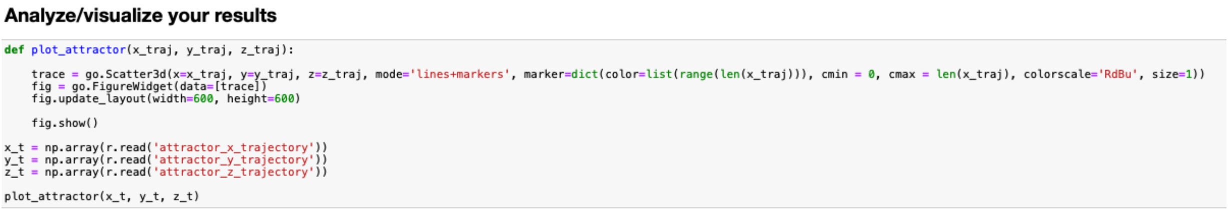
Get a list of all inputs and outputs

3. Set parameters

The Sim2Ls library checks that parameters fall within the ranges established by the developer

4. Run the Sim2L

Developing a Sim2L – Step 3: execution notebook (Part 2)



Analyze/visualize your results

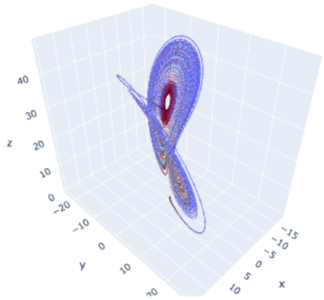
```
def plot_attractor(x_traj, y_traj, z_traj):

    trace = go.Scatter3d(x=x_traj, y=y_traj, z=z_traj, mode='lines+markers', marker=dict(color=list(range(len(x_traj))), cmin = 0, cmax = len(x_traj), colorscale='RdBu', size=1))
    fig = go.FigureWidget(data=[trace])
    fig.update_layout(width=600, height=600)

    fig.show()

x_t = np.array(r.read('attractor_x_trajectory'))
y_t = np.array(r.read('attractor_y_trajectory'))
z_t = np.array(r.read('attractor_z_trajectory'))

plot_attractor(x_t, y_t, z_t)
```

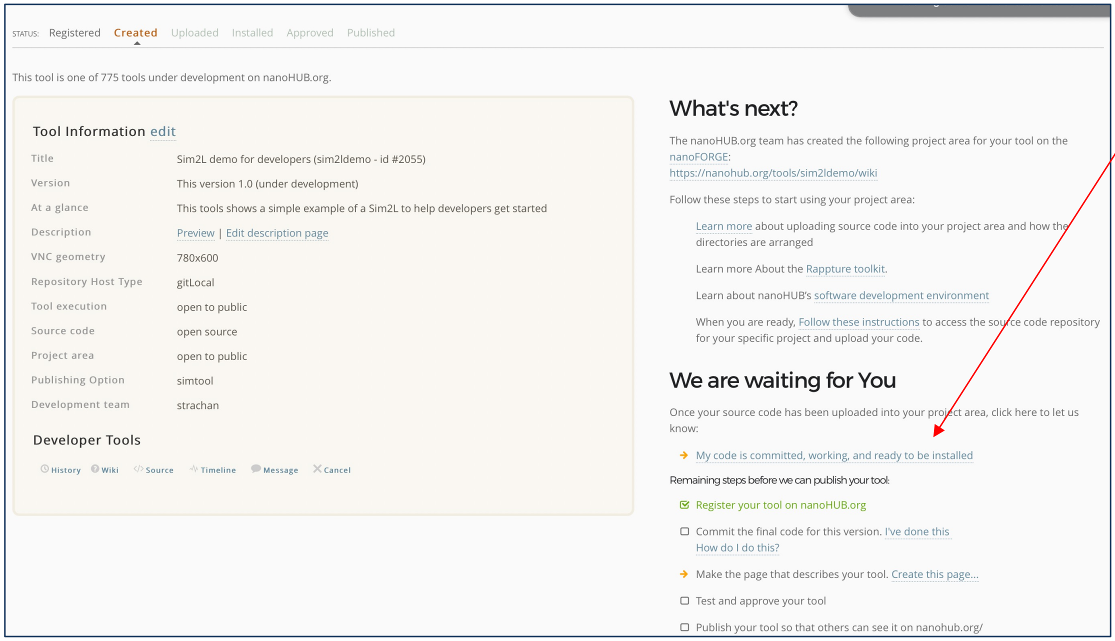


5. Get results, analyze, and visualize

Publishing a Sim2L

When you are done, just let the nanoHUB team know

https://nanohub.org/tools/sim2ldemo/status


STATUS: Registered Created Uploaded Installed Approved Published

This tool is one of 775 tools under development on nanoHUB.org.

Tool Information edit

Title	Sim2L demo for developers (sim2ldemo - id #2055)
Version	This version 1.0 (under development)
At a glance	This tools shows a simple example of a Sim2L to help developers get started
Description	Preview | Edit description page
VNC geometry	780x600
Repository Host Type	gitLocal
Tool execution	open to public
Source code	open source
Project area	open to public
Publishing Option	simtool
Development team	strachan

Developer Tools

What's next?

The nanoHUB.org team has created the following project area for your tool on the nanoFORGE:
https://nanohub.org/tools/sim2ldemo/wiki

Follow these steps to start using your project area:

Learn more about uploading source code into your project area and how the directories are arranged

Learn more About the Rappture toolkit.

Learn about nanoHUB's software development environment

When you are ready, Follow these instructions to access the source code repository for your specific project and upload your code.

We are waiting for You

Once your source code has been uploaded into your project area, click here to let us know:

→ My code is committed, working, and ready to be installed

Remaining steps before we can publish your tool:

☑ Register your tool on nanoHUB.org

□ Commit the final code for this version. I've done this
How do I do this?

→ Make the page that describes your tool. Create this page...

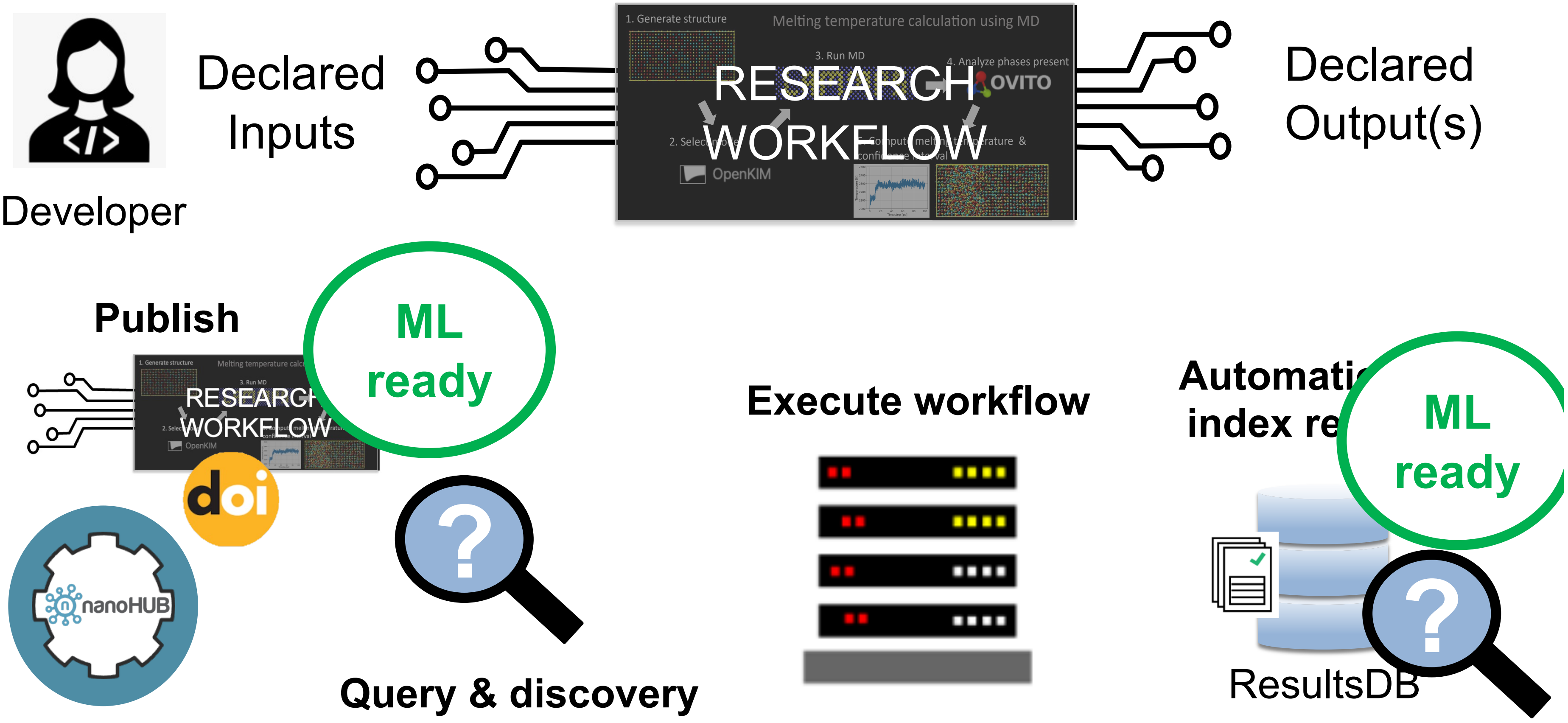
□ Test and approve your tool

□ Publish your tool so that others can see it on nanohub.org/

Click here

The nanoHUB team will stage the tool and you will be able to test it and make changes if needed before publication

Workflow and data are FAIR and ML-ready



Declared Inputs

Developer

Declared Output(s)

Publish

ML ready

Execute workflow

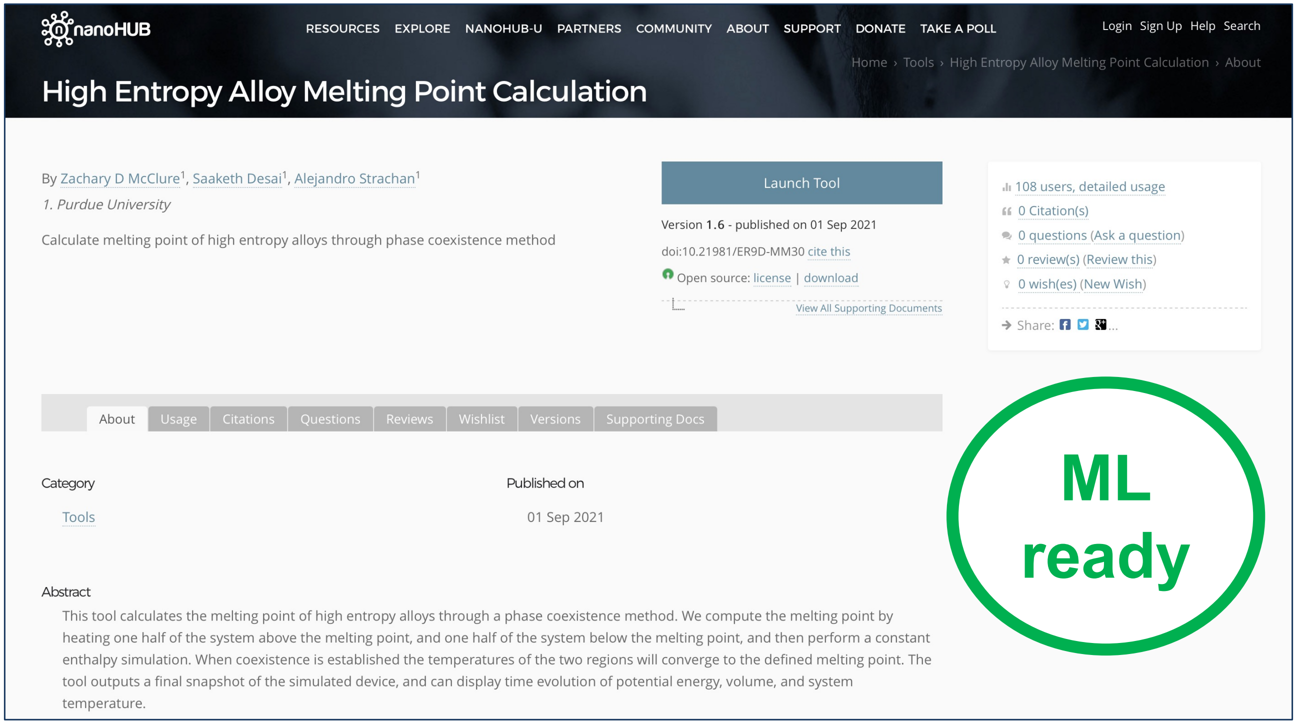
Automatic index re

ML ready

Query & discovery

ResultsDB

Sim2L example: MD simulations of melting temperature



nanoHUB
RESOURCES EXPLORE NANOHUB-U PARTNERS COMMUNITY ABOUT SUPPORT DONATE TAKE A POLL
Login Sign Up Help Search
Home › Tools › High Entropy Alloy Melting Point Calculation › About

High Entropy Alloy Melting Point Calculation

By Zachary D McClure[1], Saaketh Desai[1], Alejandro Strachan[1]

1. Purdue University

Calculate melting point of high entropy alloys through phase coexistence method

Version 1.6 - published on 01 Sep 2021

doi:10.21981/ER9D-MM30 cite this

Open source: license | download

View All Supporting Documents

108 users, detailed usage
0 Citation(s)
0 questions (Ask a question)
0 review(s) (Review this)
0 wish(es) (New Wish)

Category

Tools

Published on

01 Sep 2021

Abstract

This tool calculates the melting point of high entropy alloys through a phase coexistence method. We compute the melting point by heating one half of the system above the melting point, and one half of the system below the melting point, and then perform a constant enthalpy simulation. When coexistence is established the temperatures of the two regions will converge to the defined melting point. The tool outputs a final snapshot of the simulated device, and can display time evolution of potential energy, volume, and system temperature.

Inputs:

- Alloy composition
- Simulation details

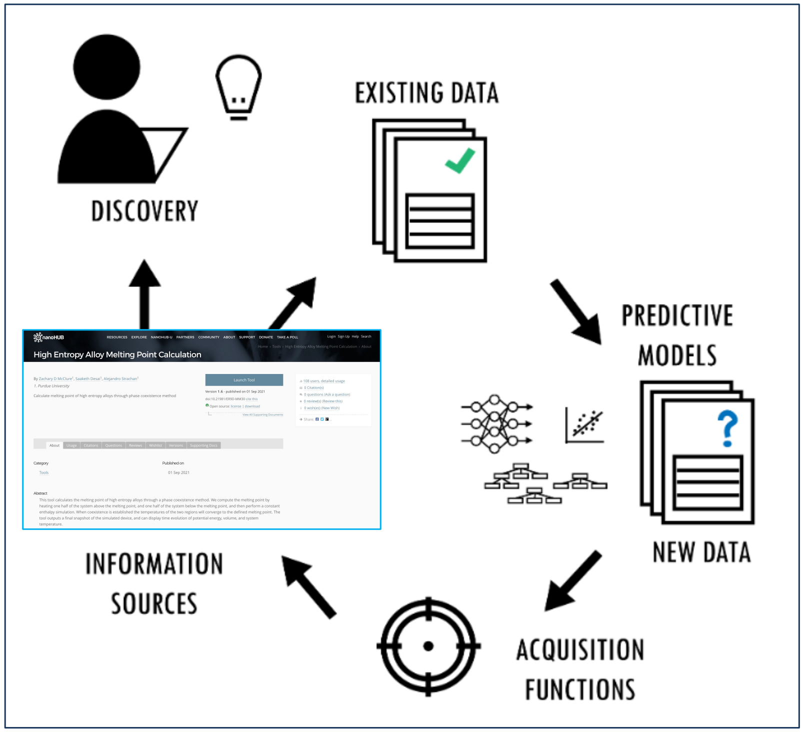
Outputs:

- Convergence
- Predicted melting temp
- Confidence interval

https://nanohub.org/tools/meltheas

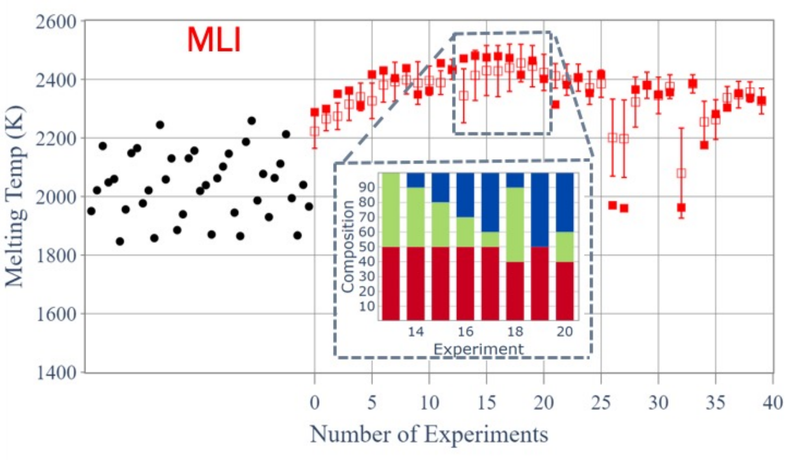
Sim2L example: launching a Sim2L from an AI/ML workflow

Autonomous ML workflow driving physics-based sims



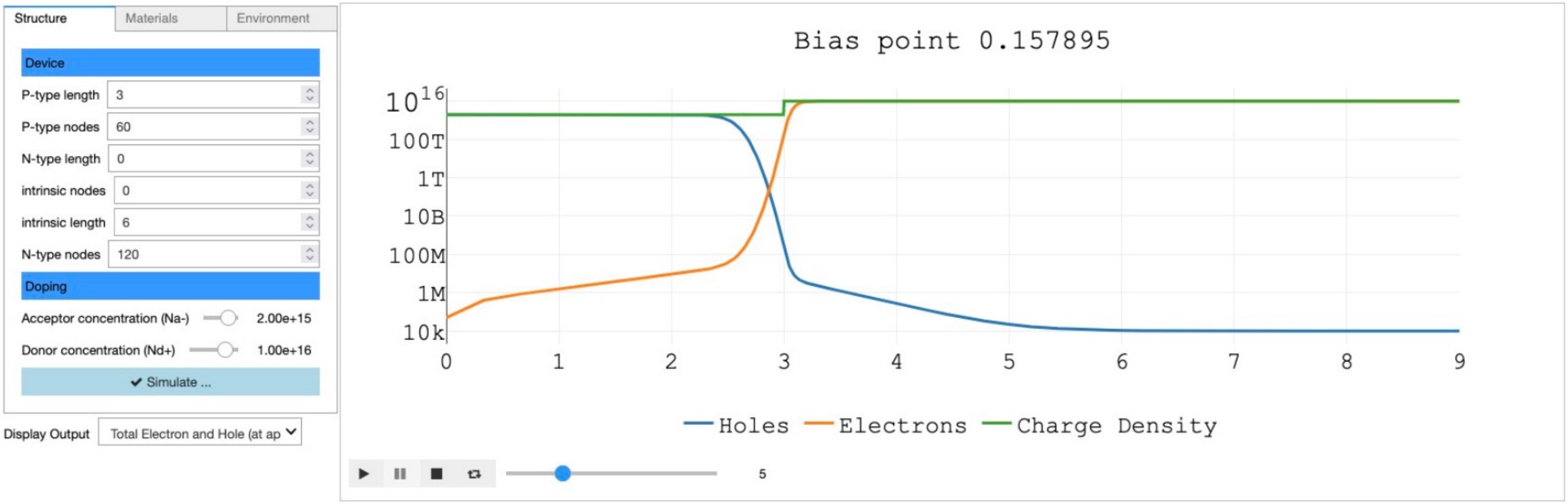
Find a high-entropy alloy with the highest melting temperature

~15 simulations out of 544



https://nanohub.org/tools/activemeltheas

Sim2L example: an App running a Sim2L



Device

P-type length: 3

P-type nodes: 60

N-type length: 0

intrinsic nodes: 0

intrinsic length: 6

N-type nodes: 120

Doping

Acceptor concentration (Na-): 2.00e+15

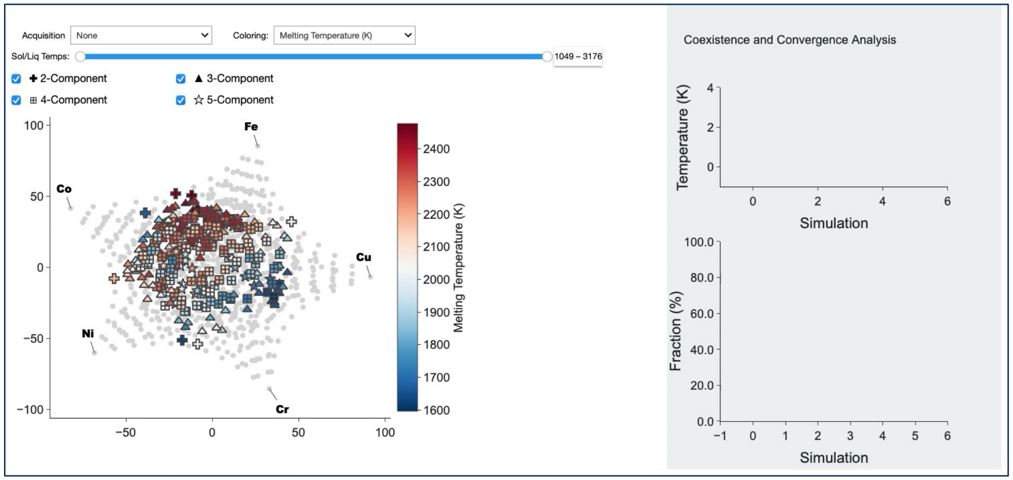
Donor concentration (Nd+): 1.00e+16

Display Output: Total Electron and Hole (at ap

https://nanohub.org/tools/st4pnjunction

Exploring the ResultsDB

Every successful Sim2L run from published tools is indexed in the ResultsDB



Acquisition: None

Coloring: Melting Temperature (K)

Sol/Liq Temps: 1049 – 3176

☑ ✚ 2-Component

☑ ▲ 3-Component

☑ ⊞ 4-Component

☑ ☆ 5-Component

nanoHUB.org/tools/meltdashboard

This App explores the runs of the meltHEAs Sim2L

Temperatures are plotted in a 2D representation of the 5-element space

Panel on the right shows the individual results for each alloy

Documentation on the ResultsDB available at: https://nanohub.org/developer/api/endpoint/dbexplorer

Additional resources

- Sim2Ls: FAIR simulation workflows and data Martin Hunt, Steven Clark, Daniel Mejia, Saaketh Desai, Alejandro Strachan. PLOS ONE. 2022 Mar 10;17(3):e0264492. https://doi.org/10.1371/journal.pone.0264492

- Documentation: https://simtool.readthedocs.io/en/stable/

- Additional information with hands-on information: https://youtu.be/7KHwJdJwtxc