Teaching AI to Speak LAMMPS: A Practical Guide to Prompting and Script Checking

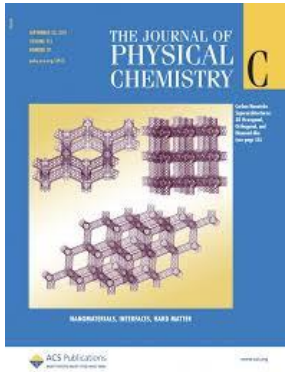
Ethan Holbrook, Juan C. Verduzco, and Alejandro Strachan

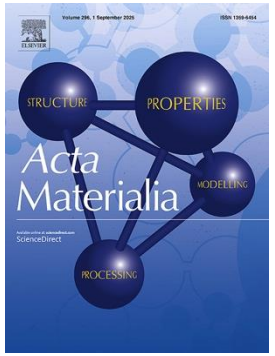
Purdue University

LAMMPS Users Meeting, 2025

The future of computational science

One paper, one code





AI and automation

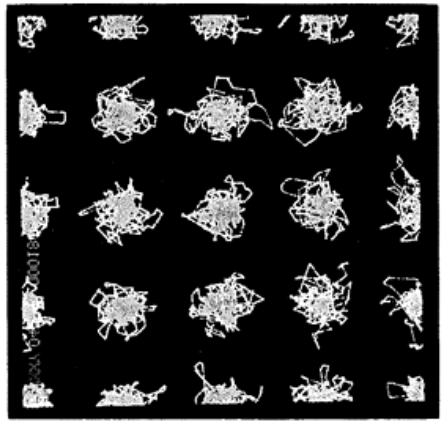
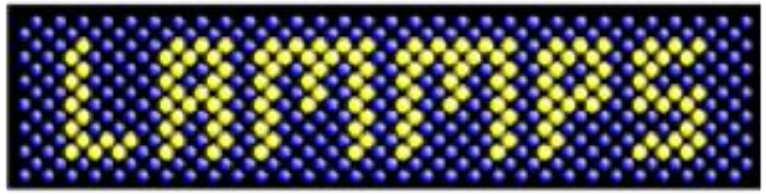


FIG. 2. The traces of 32 hard sphere particles in the periodic boundary conditions in the solid phase for about 3000 collisions.

B. J. Alder; T. E. Wainwright



MD code simulation software

The process of writing a script

How would you start the task of running a simulation?

1. Understand the physics

- time/length scales of relevant phenomena
- thermodynamic ensemble
- potential
- deformation, thermostat, barostat

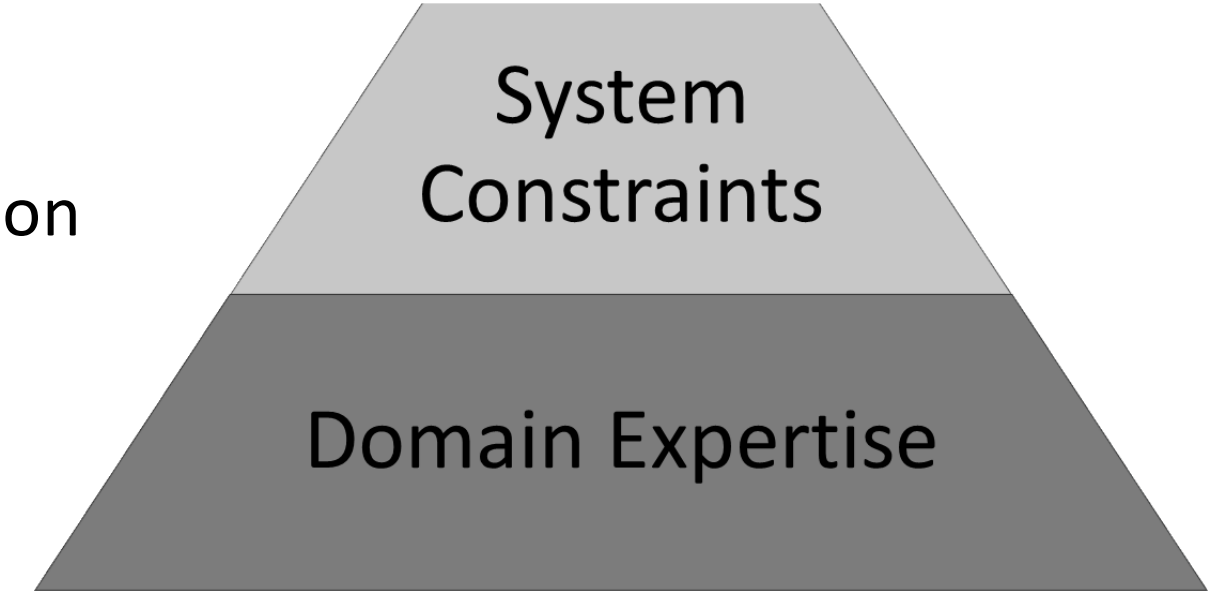
The process of writing a script

How would you start the task of running a simulation?

2. Design an experiment / simulation

1. Understand the physics

- boundary conditions
- neighbor list specs
- data output
- spatial definitions



The process of writing a script

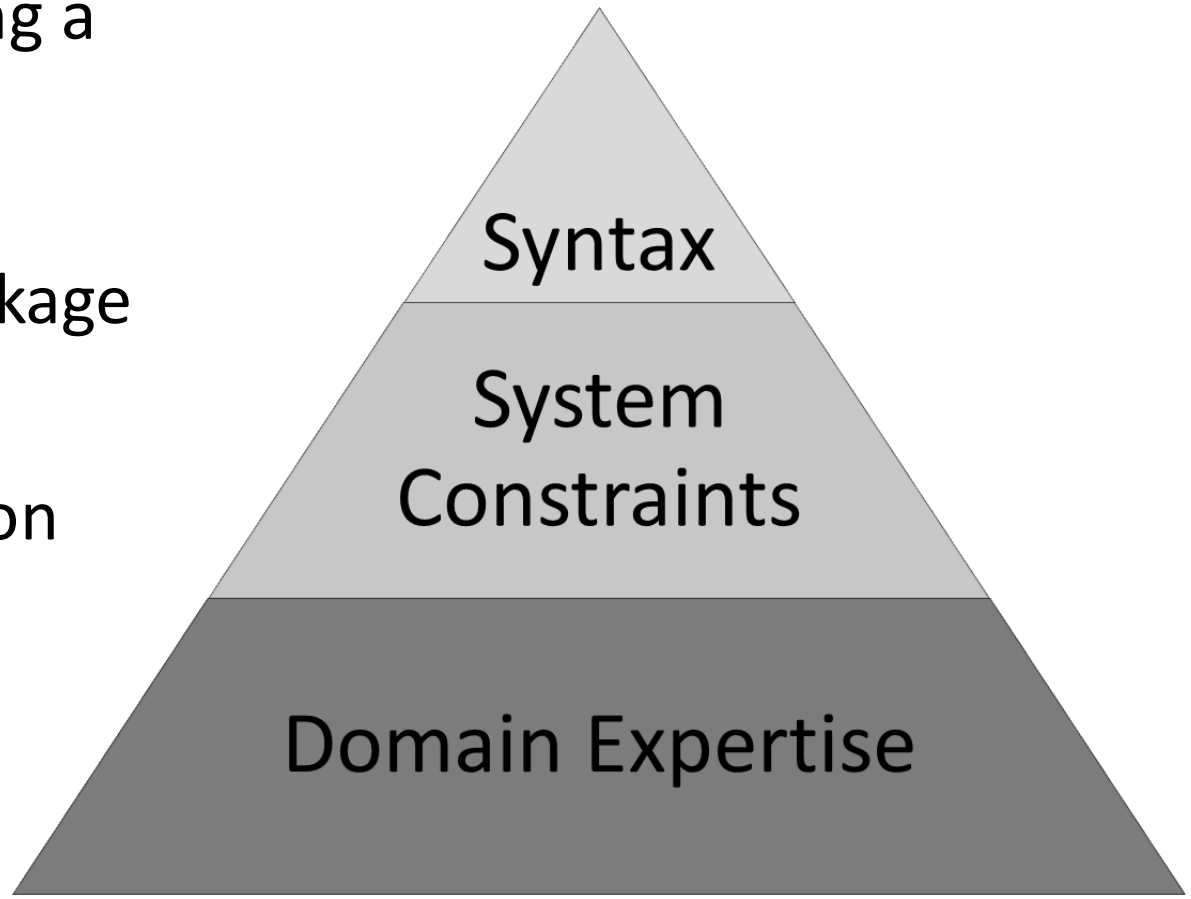
How would you start the task of running a simulation?

3. Implement it in a simulation package

2. Design an experiment / simulation

1. Understand the physics

- fix, run, timestep, etc
- command ordering
- units



Talking to the software packages

- Domain-specific languages
- Users require a lot of experience with syntax / documentation
- No compiler to catch malformed logic
- No linter to warn about misplaced fix commands
- No way to check input file syntax before execution

Category	General-Purpose Languages	Scientific DSLs
Syntax Checking	Compilers, Linters	Often manual
Flexibility vs Rigidity	Abstracts complexity	Requires some strict ordering
Error Feedback	IDE stack traces, warnings	Runtime crashes

Large Language Models

- Broad knowledge
- Few-shot learners
- Excellent at NLP tasks
- Fluent and coherent

- Prone to hallucinations
- Inaccessible information sources
- Poor structured reasoning
- Limited task generalization

How do we address these limitations? Context through prompt engineering & tools

Prompt Engineering

A prompt is an instruction provided to an LLM.

Specific, logical, and clear prompts are crucial to get good answers.

Write me a LAMMPS script for a simulation with atoms.

No clear instruction

Missing context

Vague

Generate a complete LAMMPS input script that simulates a 5 nm × 5 nm × 10 nm FCC copper nanowire under uniaxial tensile strain at 300 K. Use an EAM potential for Cu, with an microcanonical ensemble and apply a constant engineering strain rate of 1×10^{8} s^{-1} until 20% elongation. Output atomic coordinates every 1 ps and stress–strain data to a text file.

- System geometry and scale
- Material and potential type
- Ensemble
- Loading conditions
- Output requirements

**Image generated with DALLE-3

Prompt Engineering for LAMMPS Scripts

You are an expert in molecular dynamics simulations and LAMMPS scripting

Roleplaying

Your task is to generate complete, runnable LAMMPS input scripts based only on the provided method description. Do not display intermediate thoughts, summaries, or explanations—only output the final script.

Clear instruction

- *Explicitly define all commands, even if using the defaults.*
- *Assume LAMMPS version Jul2025*
- *Make sure to use the potential: NiAl.eam.alloy*

Context awareness

Generate a LAMMPS script using the same format as the following scripts:

However, set it up for tungsten instead.

Providing Examples (Few-shot)

Prompt Engineering for LAMMPS Scripts

You must follow strict formatting rules and proceed step by step, reasoning through each section logically before writing the final output. Plan the LAMMPS input script structure, using the following clearly labeled sections:
Initialization, System Construction, Potential, Miscellaneous (if needed), Production Run

Generate each section by:

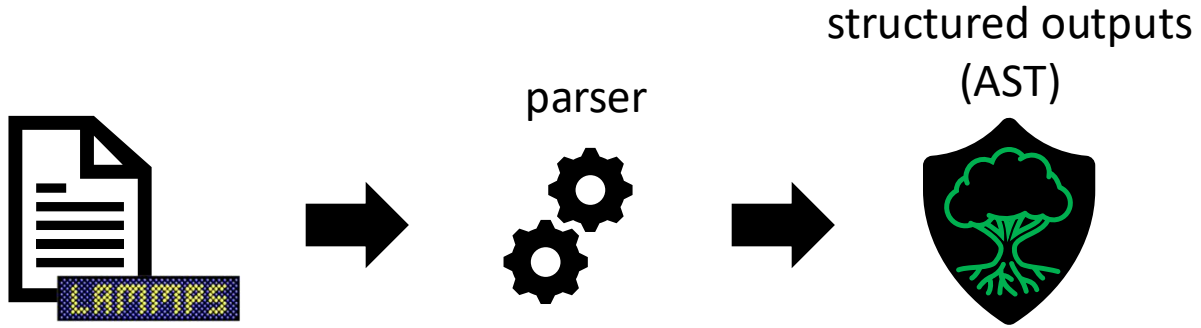
- *Determining what LAMMPS commands are required based on the method description.*
- *Use default values only when the method explicitly mentions them or when required by LAMMPS syntax but still write them explicitly.*
- *Ensure syntax correctness and functional completeness.*

Step-by-step logic (Chain-of-thought)

Parser to catch common syntax errors

- Parser tools catch syntax errors by translating DSLs into more structured forms

- Having a parser leads to:
  - faster development
  - easier debugging
  - overall better user experience



- ‘Lark’ package, widely used in python.

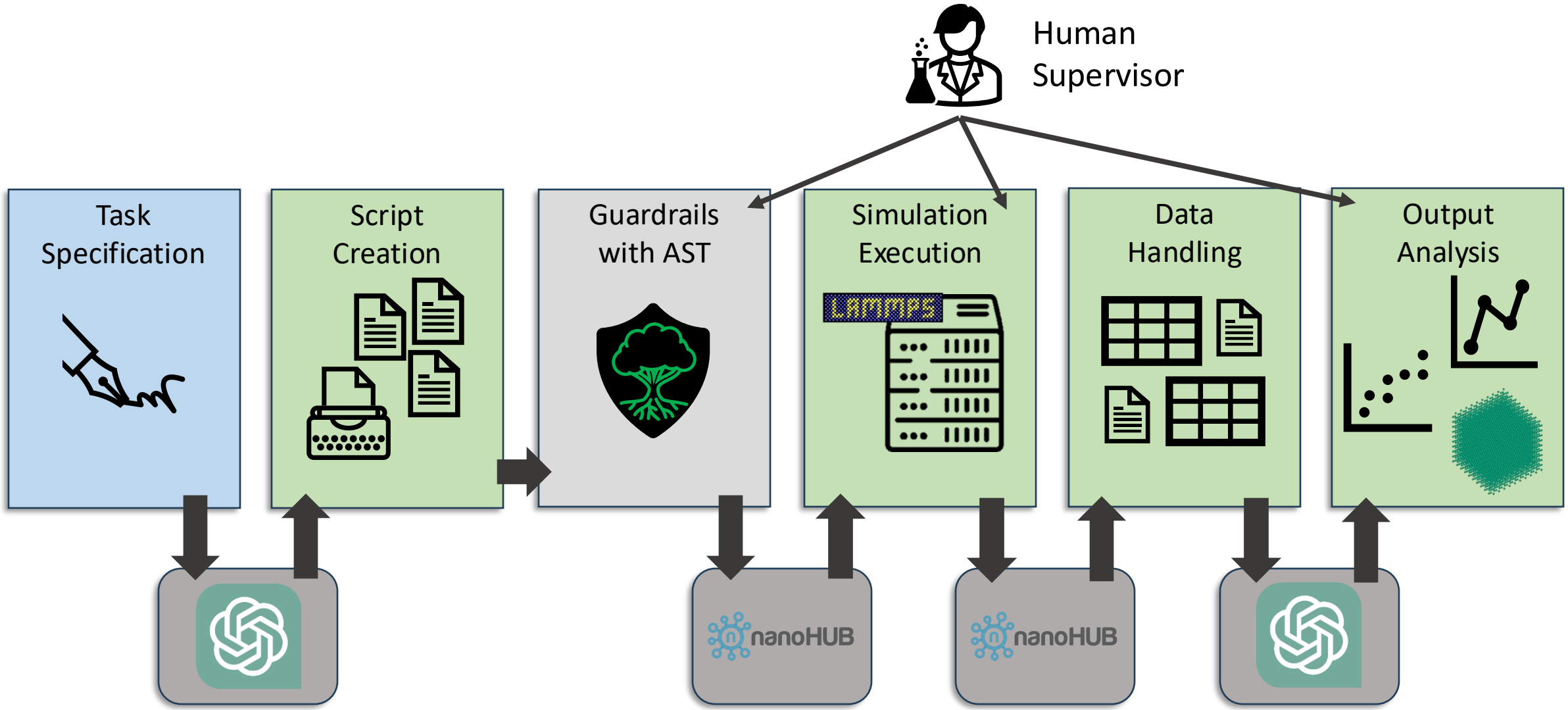
- Current parser status
  - 50 most common commands
  - One version of LAMMPS

Lark - a parsing toolkit for Python

Lark is a parsing toolkit for Python, built with a focus on ergonomics, performance and modularity.

Lark can parse all context-free languages. To put it simply, it means that it is capable of parsing almost any programming language out there, and to some degree most natural languages too.

Autonomous workflow dreams



Evaluating LLM generated scripts

Sanitizer

Parser/Linter

Executor

Accuracy Checker

Most common errors:

- Mismatched syntax in pair styles (eam vs. eam/alloy vs. eam/fs)
- Region/group specifications
- Wrong units

nanoHUB - Creating an account

"An engineering gateway accelerating scientific discovery through simulation, education, and AI-powered tools – all in the cloud"

320k registered users
(25,837 simulation users)

8,262 resources
(teaching materials, lecture notes, homework problems)

866
published tools

3,000 citations
(h-index 135)

Founded in 2002, as open and free online platform for computational education, research, and collaboration in nanotechnology, materials science, and related fields.

Register at: https://nanohub.org/register/

Simple ways to use LLMs

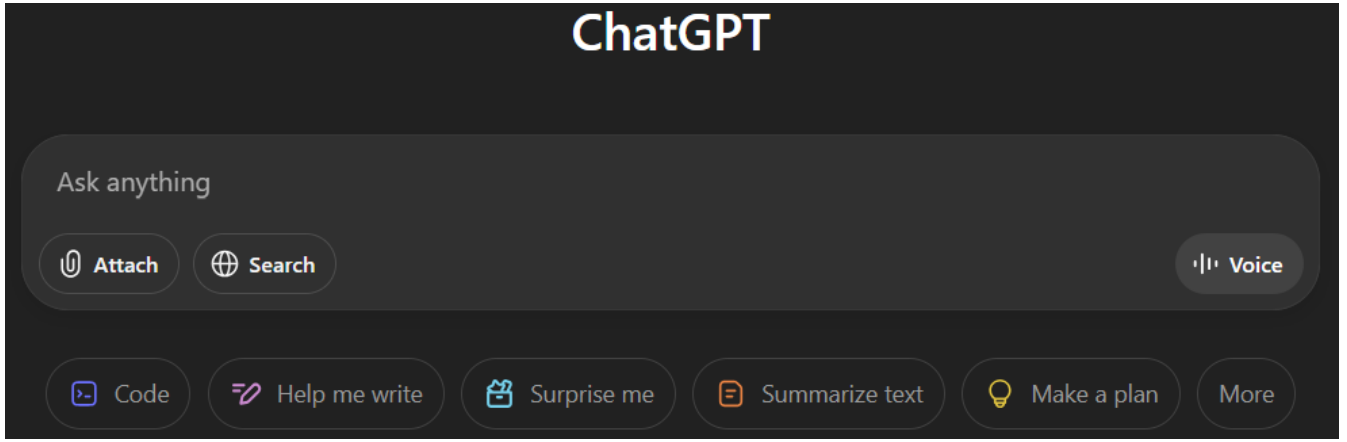
For this demonstration we will require a way to generate scripts.

Options:

1) Implemented in the tool using an OPENAI API Key: https://platform.openai.com/api-keys → Ex: sk-...gVkA
2) Any of your favorite LLMs (If you've used any)
3) Bypass version using the free version of Chat GPT

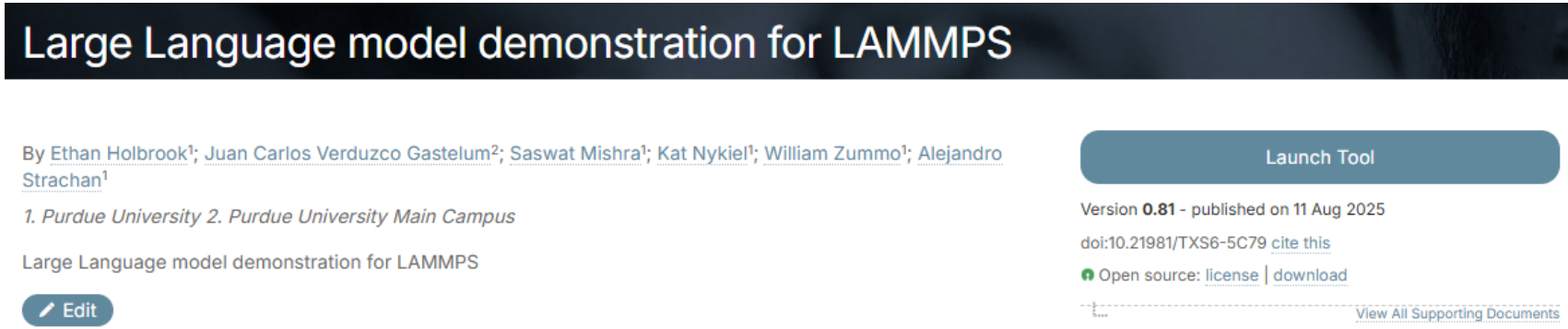
Favorite LLM

Free version of ChatGPT @ chatgpt.com



Today's exercise

Keep the run count/system size low so that you can get quick results.



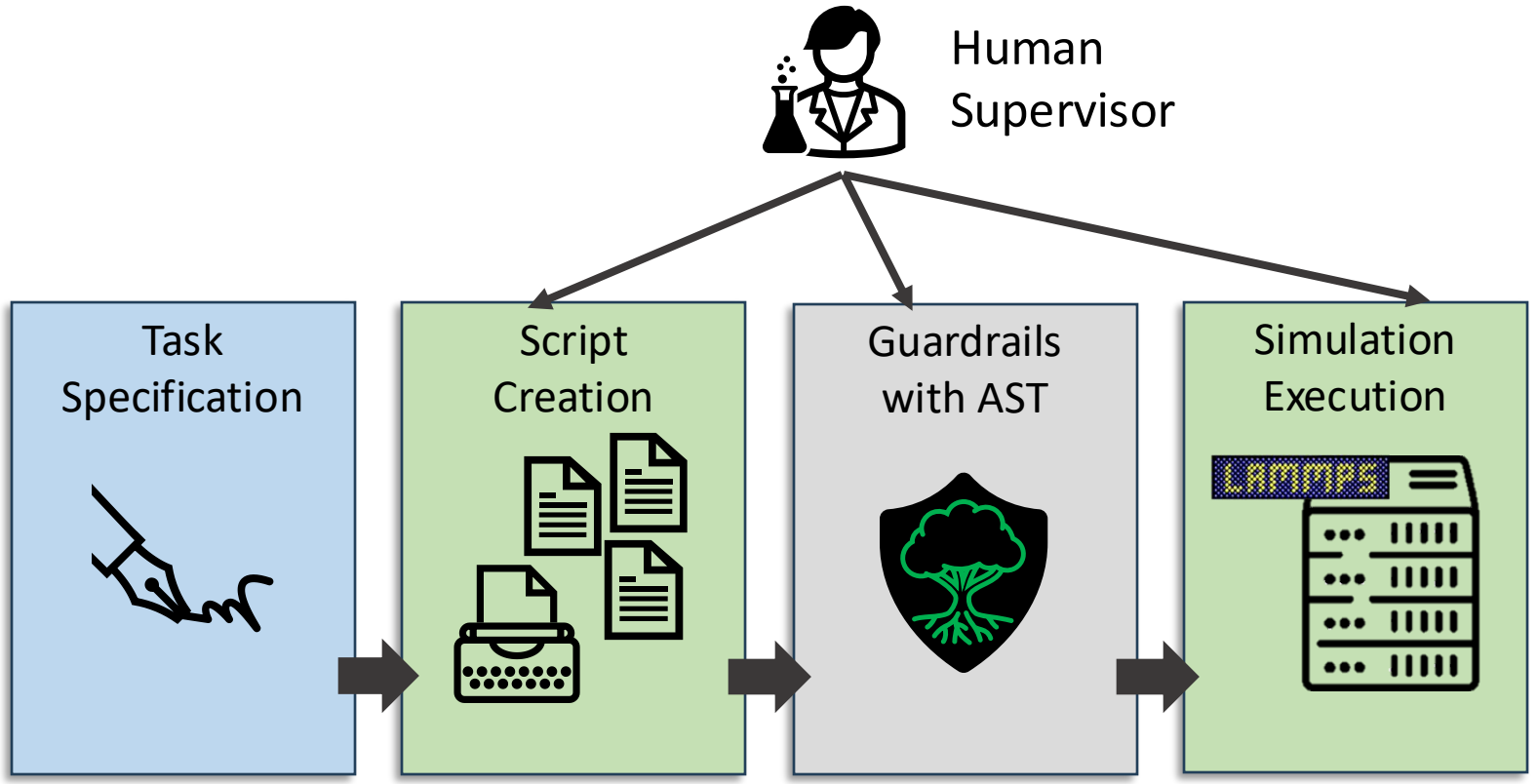
Your tasks:

1. Go to the URL below
2. Find an OpenAI Api Key/use bypass with LLM of choice
3. Check out the parser!
4. Try changing some script parameters up
5. Run! (could take a while) ~5 minutes **(limit run value)**

6. (Optional) Get results by downloading the .yaml file
7. (Optional) Try generating a methods section!

https://nanohub.org/tools/llm4lammps

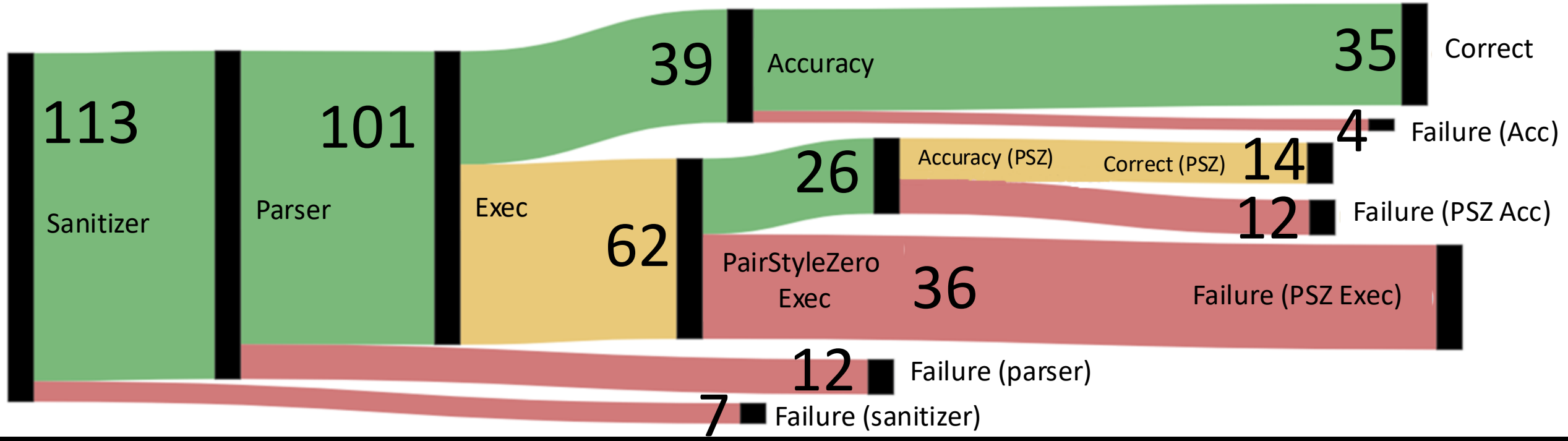
Live examination of your LLMs



Collecting Stats/Live Sankey

Our experience with model performance

- **4 models:** GPT-4o, GPT-o3, GPT-4.1, Claude Opus 4
  - **3 Prompts:** Simple, Medium, Complex
- Careful of misleading word-matching
- Region/group specifications, units, and pair style most prevalent issues



Conclusions

- LLMs are ready for this application space, still quite useful
- These tools are useful as collaborators / copilots and should never replace people knowing their documentation and how to debug scripts. Not everything you read from an LLM is true!
- Looking for collaborators

Contributing LARK - Activity

- Another activity – hone your ability to find / understand LAMMPS documentation

- We are currently requesting help (via PRs) on our Github to complete a version of the parser for LAMMPS.

- LARK grammar itself is somewhat technical, but they have a nice IDE for devs:
  - https://www.lark-parser.org/ide/

Current grammar in our package

Choose your favorite command and **contribute**!

Thanks for listening!

- Alejandro Strachan's Group
- Email: holbrooe@purdue.edu
- Questions??