

Using agents in scientific research

Isidora
Araya Day

6 September
2026

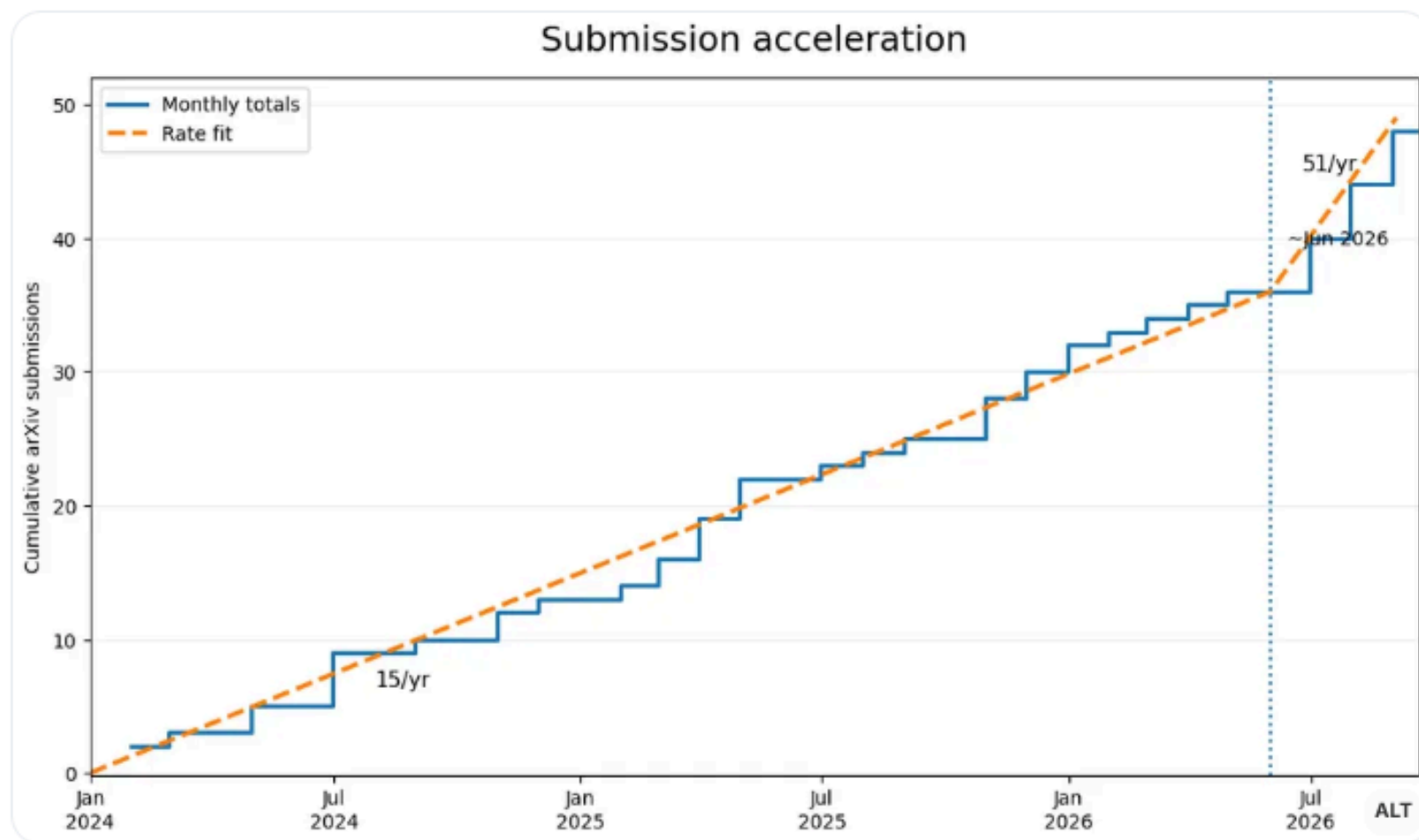


Anton Akhmerov

@antonakhmerov.org

+ Follow

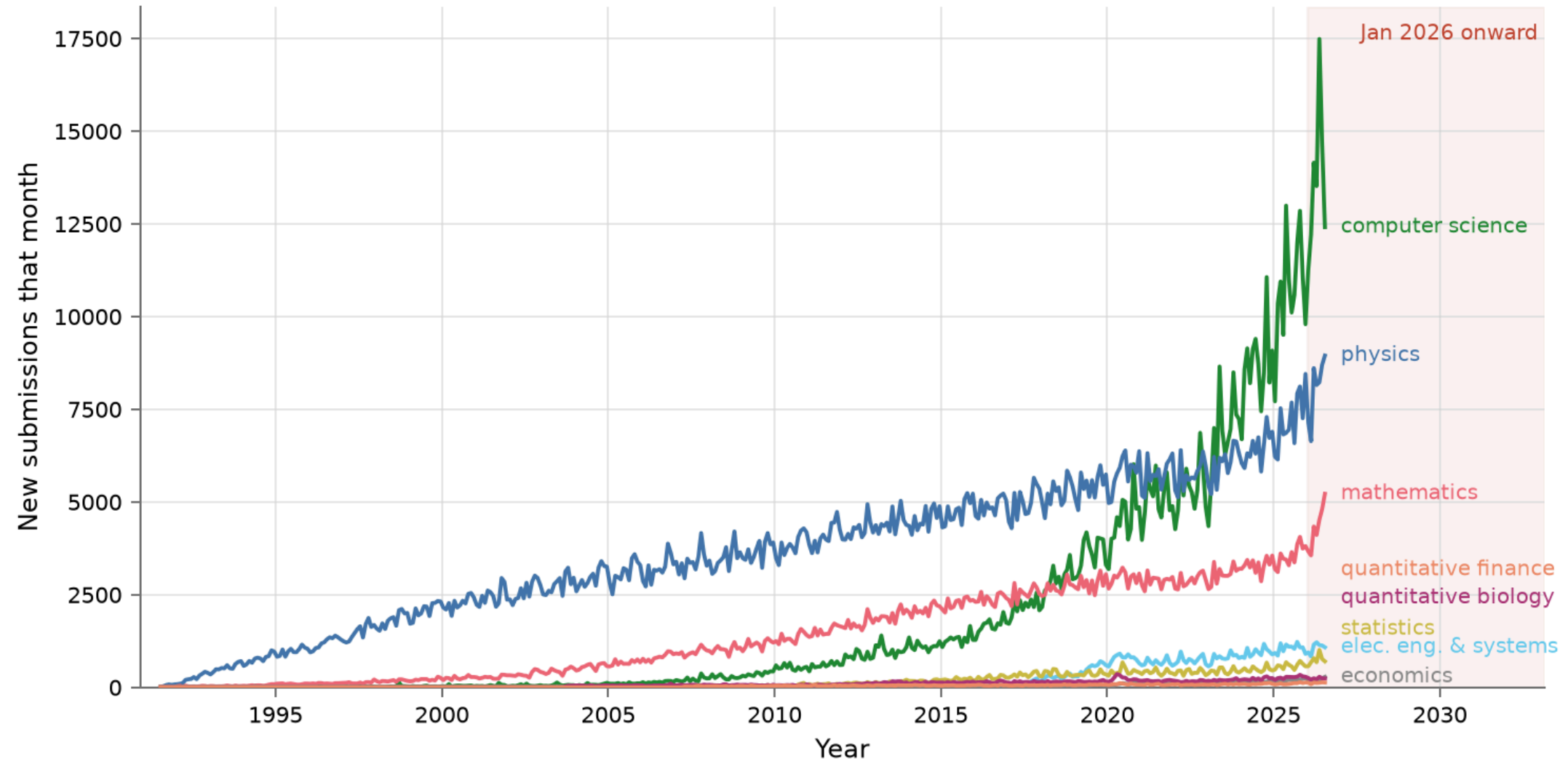
Here is monthly publication rate of a colleague. Recent papers (after the dashed line) acknowledge Claude usage, and the publication rate goes 3x. The paper quality stays roughly constant.



2:02 PM · Sep 3, 2026 🌐 Everybody can reply

arXiv submissions per month, by field

Primary category grouped to arXiv's own top level; volume, not discovery



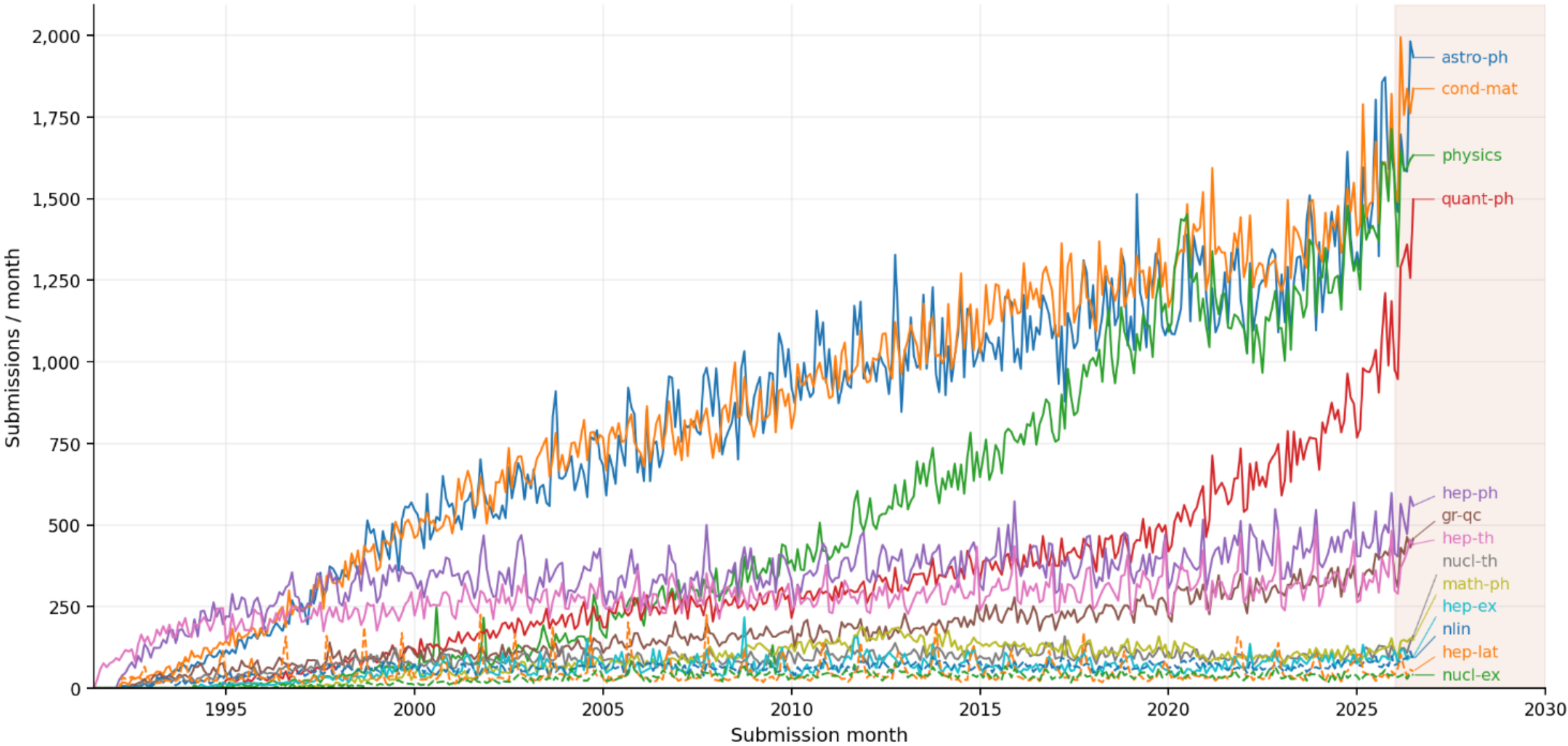
Source: arXiv OAI-PMH metadata, month of first version, primary category only.

Source: tecunningham/ai-discovery-data · output-arxiv

arXiv physics submissions per month

×

1991-07-2026-07 · primary categories only · monthly counts, without smoothing



Source: repository arXiv OAI-PMH category snapshot. Each paper counted once; cross-lists excluded. Includes math-ph and legacy archives. August 2026 is partial and excluded; this is not a live refresh.

Source: tecunningham/ai-discovery-data · output-arxiv

Inspect this repo, focus on the output-arxiv section. Collect the data and make a plot of physics submissions per month as a function of time. Show full physics aggregated submissions, and also add a curve per category.

Working for 3m

I'll inspect the `output-arxiv` data, check how categories and dates are recorded, and build a monthly plot with the physics total and category curves.

>_ Ran 8 commands

The repo already contains monthly primary-category counts, so each paper can be counted once without duplicating cross-lists. I'll use July 1991–July 2026; August 2026 is partial in this snapshot.

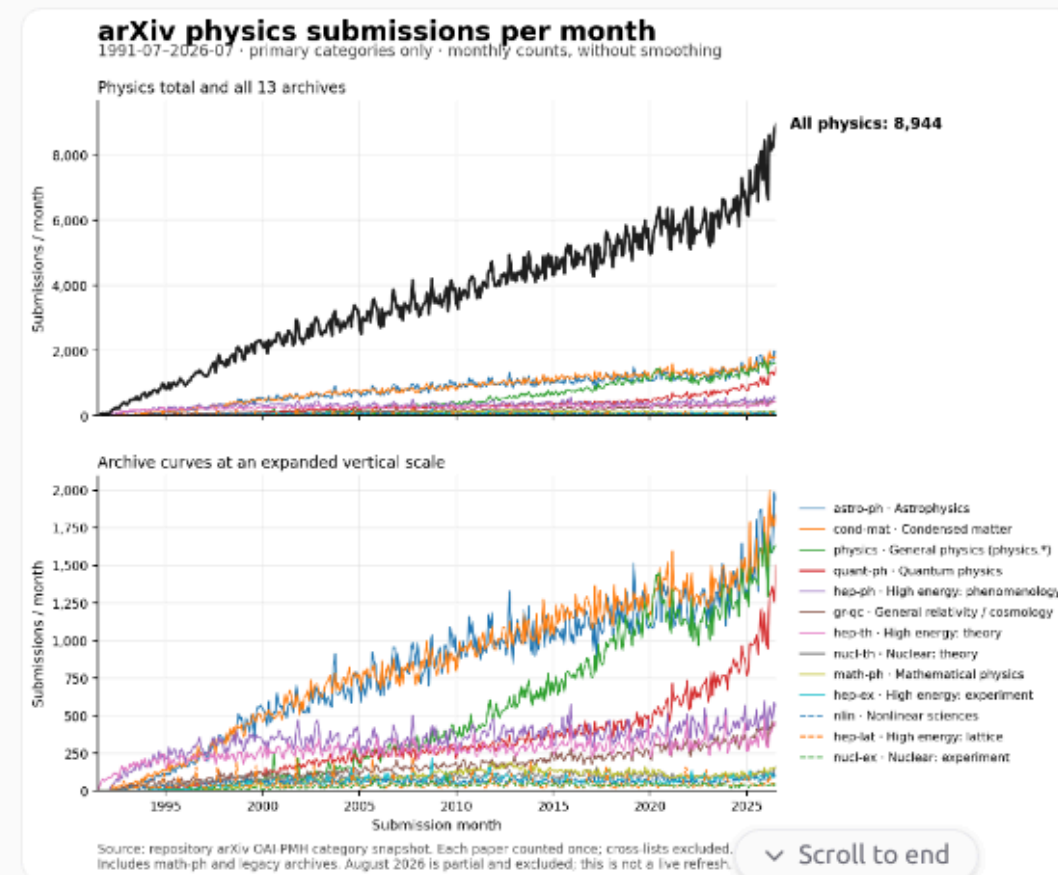
I'll plot the physics total with all 13 physics archive curves, and include a separate breakdown of their subcategories. Docker isn't installed, so I'll generate standalone plots with the local renderer and leave the repo's existing figures intact.

 `ai-discovery-data/problems/output-arxiv/analyze_physics.py`

Inspect this repo, focus on the output-arxiv section. Collect the data and make a plot of physics submissions per month as a function of time. Show full physics aggregated submissions, and also add a curve per category.

Worked for 4m 14s >

Created the plot from the repo's existing snapshot: **July 1991–July 2026**, with the physics total and all **13 physics archives**. Each paper counts once by primary category; partial August 2026 is excluded.



September 8, 2026 Applied AI

How GPT-5.6 Sol helps run quantum computing experiments

Connecting GPT-5.6 Sol to laboratory software to run and refine routine measurements on quantum chips freed Beatriz Yankelevich to focus on experiment design and data analysis.

[Read the technical case study ↗](#)[▶ Listen to article](#) 5:22[↗ Share](#)

Quantum computing is an emerging technology that uses the unique properties of quantum mechanics to process information. It could one day better simulate complex materials and molecules. Unlike conventional processors, quantum processors are built with quantum bits, or qubits. Preparing and running qubit experiments can take months and require hundreds to thousands of preliminary measurements—work that AI is poised to help with.

Beatriz Yankelevich, a graduate student in MIT's Engineering Quantum Systems Group

[Open live article ↗](#) · OpenAI — Quantum computing experiments

Case Study: Agentic Calibration of Superconducting Qubits

Beatriz Yankelevich^{1,2}, Eva Zhang³, JonLuca DeCaro³, Jeffrey A. Grover¹, and William D. Oliver^{1,2,4}

¹*Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

²*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

³*OpenAI, San Francisco, CA, USA*

⁴*Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

September 4, 2026

Contents

1 Introduction	1
2 How superconducting qubits work	3
3 Measurement workflow	4
4 Agentic calibration results	5
5 Implications for superconducting qubit research	9
6 Contributions	10

1 Introduction

Quantum computing is an emerging computational paradigm that encodes and processes information according to the laws of quantum mechanics. Phenomena such as superposition, interference, and entanglement allow quantum algorithms to manipulate information in ways that differ fundamentally from classical computation. For certain problems, this enables dramatic speedups over the best known classical methods, most famously Shor’s algorithm for integer factorization, while also offering a natural path toward simulating complex quantum systems in physics, chemistry, and materials science.

Superconducting circuits are a leading platform for quantum computing. Often described as “artificial atoms,” superconducting qubits are integrated circuits whose discrete energy levels encode quantum information. At MIT, the Engineering Quantum Systems Group (EQuS) researches superconducting quantum systems across the stack, from new fabrication techniques and qubit designs to many-qubit experiments that explore quantum error correction and emergent quantum phenomena.

Designing and executing a superconducting-qubit experiment employs a diverse skill set that draws from various scientific and engineering disciplines. The work can be roughly grouped into five categories: analytical and numerical modeling of quantum systems, chip design, chip fabrication, cryogenic and microwave engineering, and qubit calibration and measurement. Although AI can

Abstract

I Introduction

II Vibe calibration

III Agent Skill Architecture

IV Fine-tuned large language models

V Experimental Results

VI Discussion and Conclusion

References

Supplemental Material for “Vibe Calibration: Autonomous Bring-up of a 112-Qubit Superconducting Quantum Processor by a Skill-Orchestrating Language Agent”

License: arXiv.org perpetual non-exclusive license
arXiv:2606.22376v1 [quant-ph] 21 Jun 2026

Vibe Calibration: Autonomous Bring-up of a 112-Qubit Superconducting Quantum Processor by a Skill-Orchestrating Language Agent

Huikai Xu

These authors contributed equally to this work. Beijing Key Laboratory of Fault-Tolerant Quantum Computing, Beijing Academy of Quantum Information Sciences, Beijing 100193, China

Jiaxiu Han

These authors contributed equally to this work. Beijing Key Laboratory of Fault-Tolerant Quantum Computing, Beijing Academy of Quantum Information Sciences, Beijing 100193, China

Shigang Ou

These authors contributed equally to this work. Beijing National Laboratory for Condensed Matter Physics, Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China
School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
DP Technology, Beijing 100080, China

Cheng Ye

These authors contributed equally to this work. Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China
School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

Zisong Shen

Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China
School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

Jing Gao

School of Fundamental Physics and Mathematical Sciences, Hangzhou Institute for Advanced Study, UCAS, Hangzhou 310024, China
Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China
School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

Yijia Wang

Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China

Tianrui Che

Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China

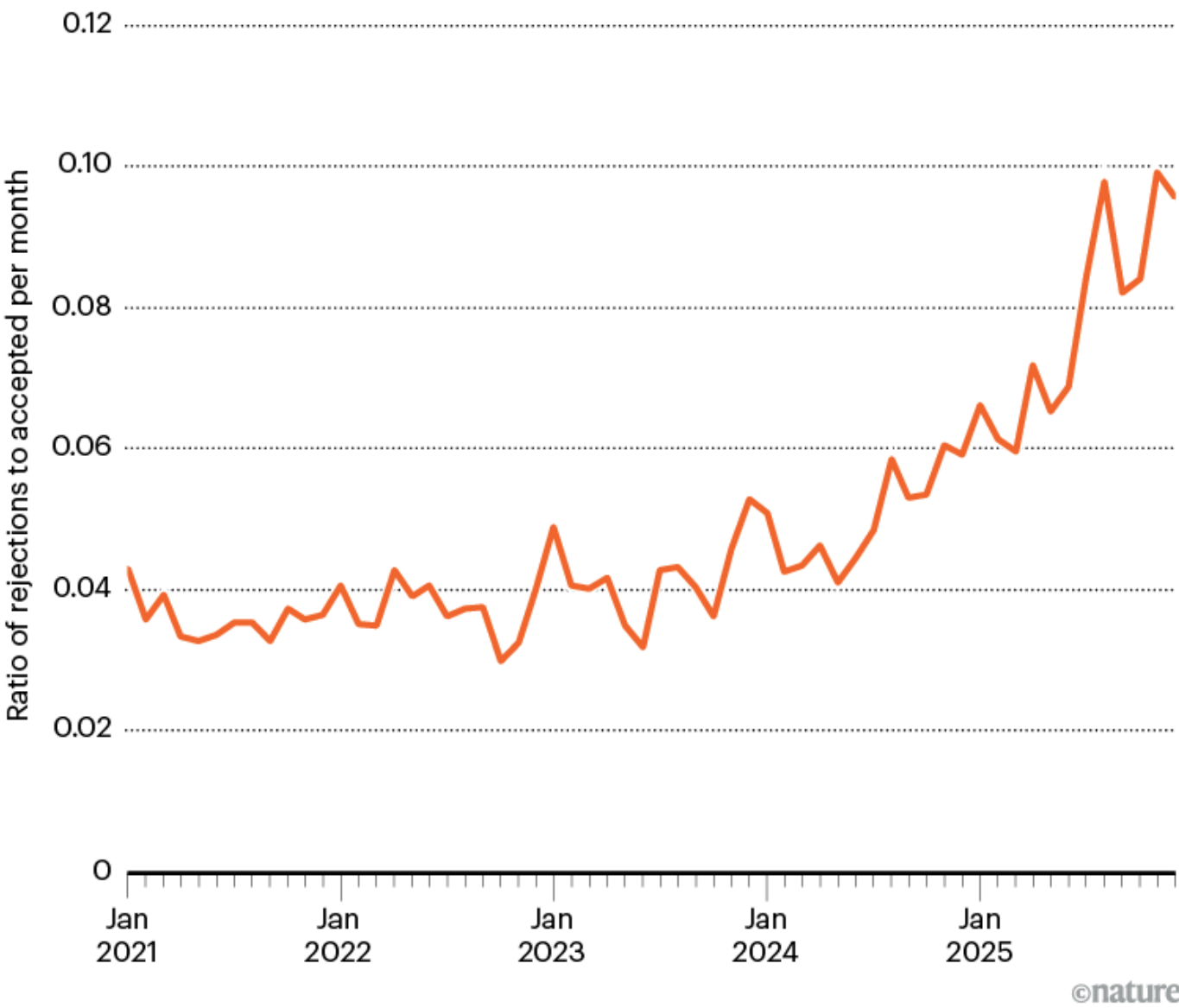
Yu Song

Beijing Key Laboratory of Fault-Tolerant Quantum Computing, Beijing Academy of Quantum Information Sciences, Beijing 100193, China

Source: Vibe Calibration — arXiv:2606.22376v1

REJECTION RATES CLIMB

The rejection rate of arXiv papers, relative to those accepted, doubled between January 2024 and 2026.



Source: Nature

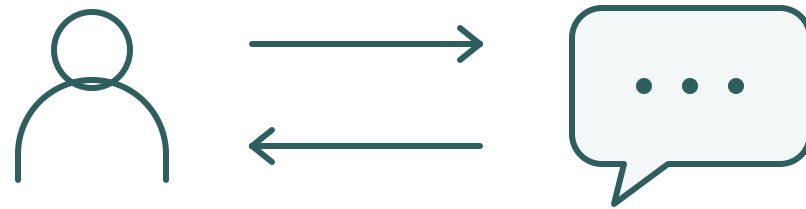
How do we continue to produce high quality research?

Four ways to work with AI

1. **Chatbot** — Ask a question and use the answer.
2. **Copilot** — Write a function together.
3. **Agent** — Give it a task.
4. **Agentic workflow** — Let it run full research loops.

Comparison between a chatbot and an agent

Chatbot

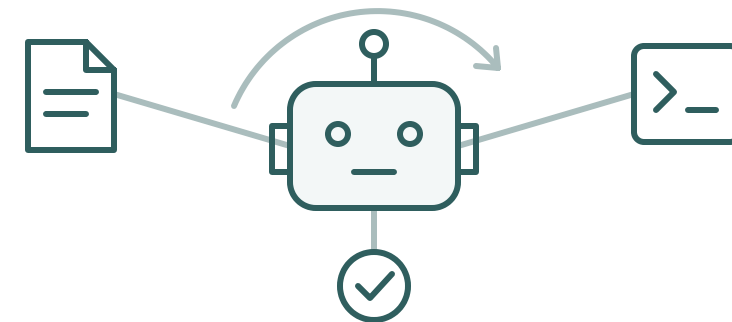


“How can I reproduce this figure?”

1. You provide context and ask a question.
2. The chatbot suggests methods or code.
3. You run the code and bring back results.

You carry out each step.

Agent



“Reproduce this figure in this project.”

1. You define the goal and checks.
2. The agent reads files, writes and runs code.
3. It checks results and revises its work.

You review the evidence and decisions.

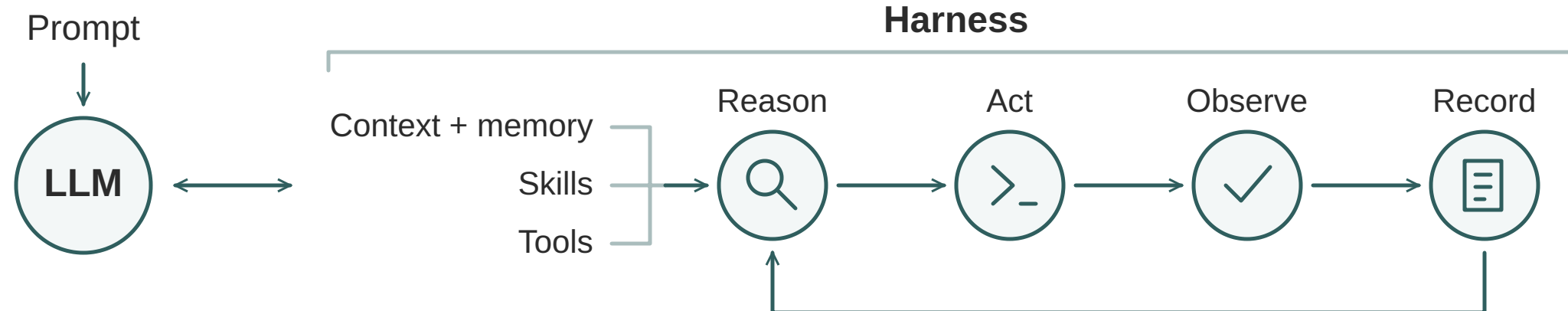
What turns an LLM into an agent?

What it needs

- **Prompt** — Your question/instruction.
- **Context** — Papers, data, notes and code.
- **Memory** — Private notes, decisions, and progress.
- **Skills** — Instructions for recurring tasks.
- **Tools** — Access to files, code execution, and tests.

How it works

1. **Reason** and inspect the project.
2. **Act** by using tools to carry out the work.
3. **Observe** the results and revise as needed.
4. **Record** the result and how to reproduce it.



Training language models to follow instructions with human feedback

OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

Demo

Tam & Kane · Probing Fermi sea topology by Andreev state transport

[arXiv:2210.08048](#) · Reproduce and extend a paper.

- 1 Create a subfolder and work there. Your job is to read [arxiv: 2210.08048](#), write a
- 2 research project codebase that reproduces the results using tight-binding simulations,
- 3 and extends the conductance simulation for a normal hamiltonian with random onsite disorder.
- 4 Report back with an html summary.

How I structure my work

ALL my projects live in Projects/

```
1 Projects/  
2 |— spin-qubit/  
3 |— another-project/  
4 |— ...
```

My workflow for every project

1. **Create** a project from my template, with its own Git repository and environment.
2. **Start Codex inside it**, with the README, notes, and a concrete task.
3. **Review** the outputs, scientific checks, and Git diff; commit the changes I accept.
4. **Resume** from saved decisions and next steps in notes/.

↓ **Explore the project:** project structure → working session

1. Organize the project

Projects / spin-qubit

```
1 spin-qubit/
2 |— .git/
3 |— README.md
4 |— AGENTS.md
5 |— pyproject.toml
6 |— pixi.lock
7 |— code/
8 |   |— spin_qubit_package/
9 |   |   |— reusable_modules/
10 |   |   |— tests/
11 |   |— simulations/
12 |   |— results/
13 |— data/
14 |— manuscript/
15 |— notes/
16 |— slides/
```

One reproducible research workspace

- **Package** — `spin_qubit_package/` holds reusable models, solvers, and plotting.
- **Scripts** — `simulations/` runs calculations; `results/` analyses saved data. Both import the package.
- **Tests** — Check known limits and catch regressions.
- **Environment** — `pyproject.toml` defines dependencies and tasks; `pixi.lock` pins versions.
- **Research record** — Save data, manuscript, notes, and slides together; document commands in the README and track changes with Git.

Repetitive task → Use a **skill** to set up a project structure!

2.Start, check, and leave a record

Start inside the project

```
1 cd ~/Projects/spin-qubit
2 codex
```

```
1 Inspect this repo, focus on the
2 tight-binding implementation.
3 Generalize the code to ...
4 Verify the result by ...
```

Process

1. **Choose a model**
2. **Give a task** — Define deliverable and checks.
3. **Ask questions** — Ask for figures, explanations.
4. **Prompt benchmarks** — Ask for tests.
5. **Ask for a record** — Write a summary.
6. **Check the diff** — If possible, inspect the Git diff.
7. **Commit** — Save the changes for later.

Prefer **focused sessions**, each with a clear goal.

3. Finalize the project

What I do:

- Discuss results with collaborators.
- Clean up the code and notes.
- Have my collaborators prompt the project.
- Disclose the use of AI.

The questions I still have:

- Should I read every line of code?
- Can another agent review it for me?
- How many checks are enough to trust the result?
- What results are worth publishing?



Producing outputs is easier than knowing whether to trust them.

What can become an agent project?

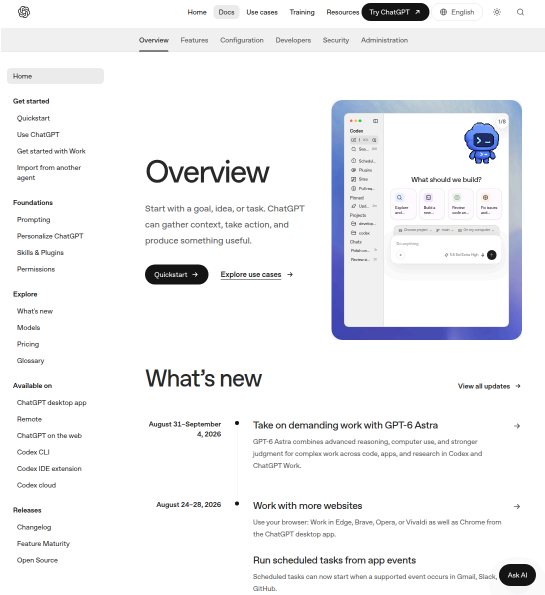
1. Creating a website
2. Filling out reimbursement forms
3. Scraping a website for data
4. Collecting literature from a chat
5. Reviewing someone else's code
6. Launching cluster simulations
7. Reproducing a figure from a paper
8. Generalizing an algorithm
9. Testing an idea

A project can be **anything with inputs and a deliverable.**

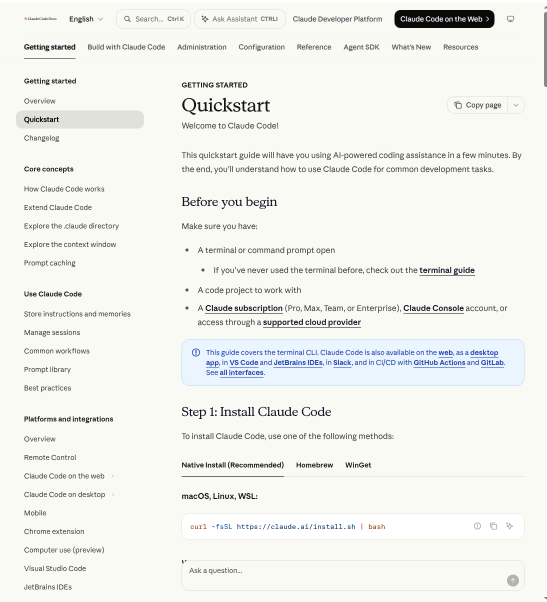
How to get started

Install and connect

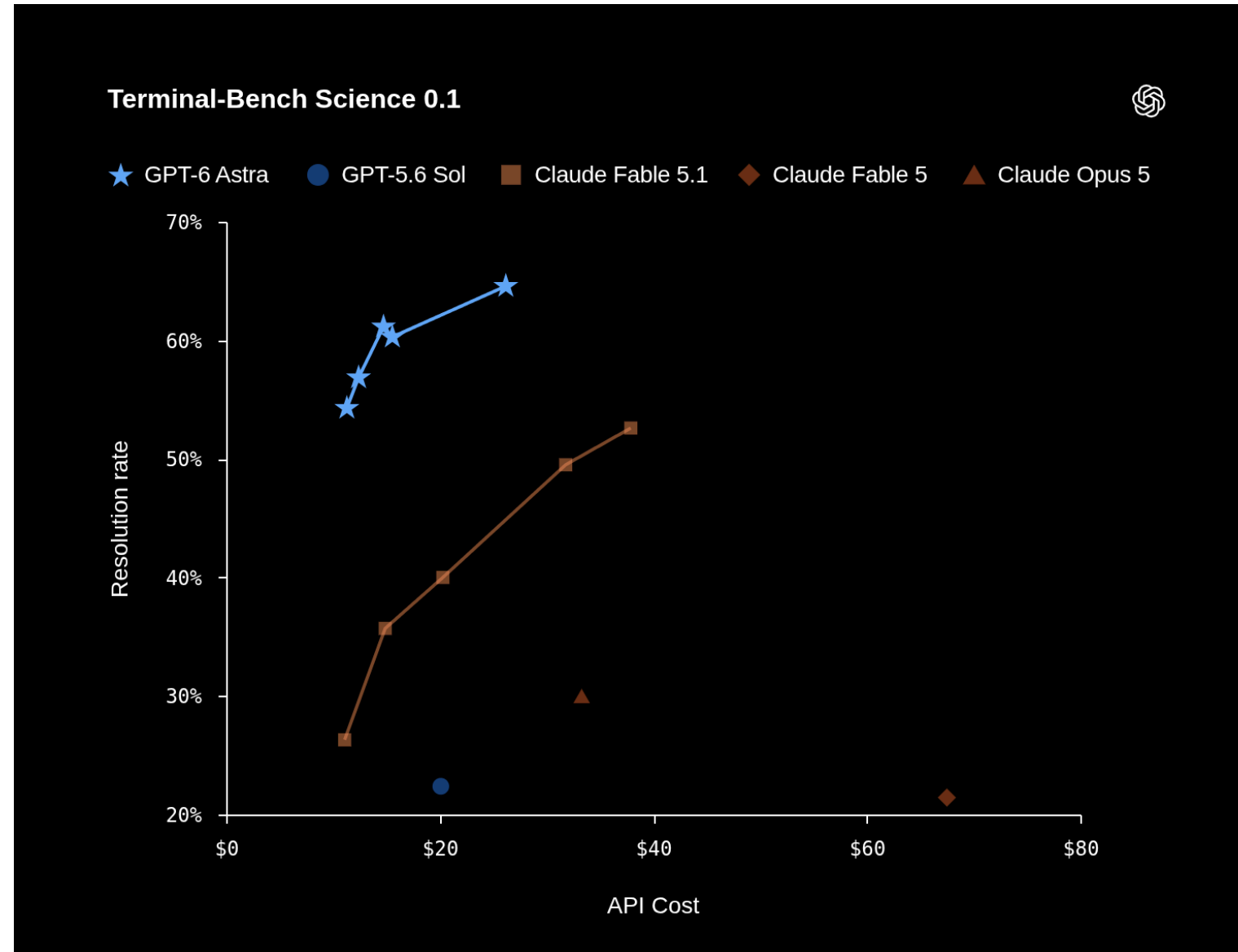
1. **Choose an interface** — An agent app or terminal CLI.
2. **Sign in** — Pay a suscription.
3. **Prepare your tools** — Ask the agent to install Git, Python, Pixi, LaTeX, etc.
4. **Open a project folder** — Use a skill to organize it.



OpenAI documentation ↗

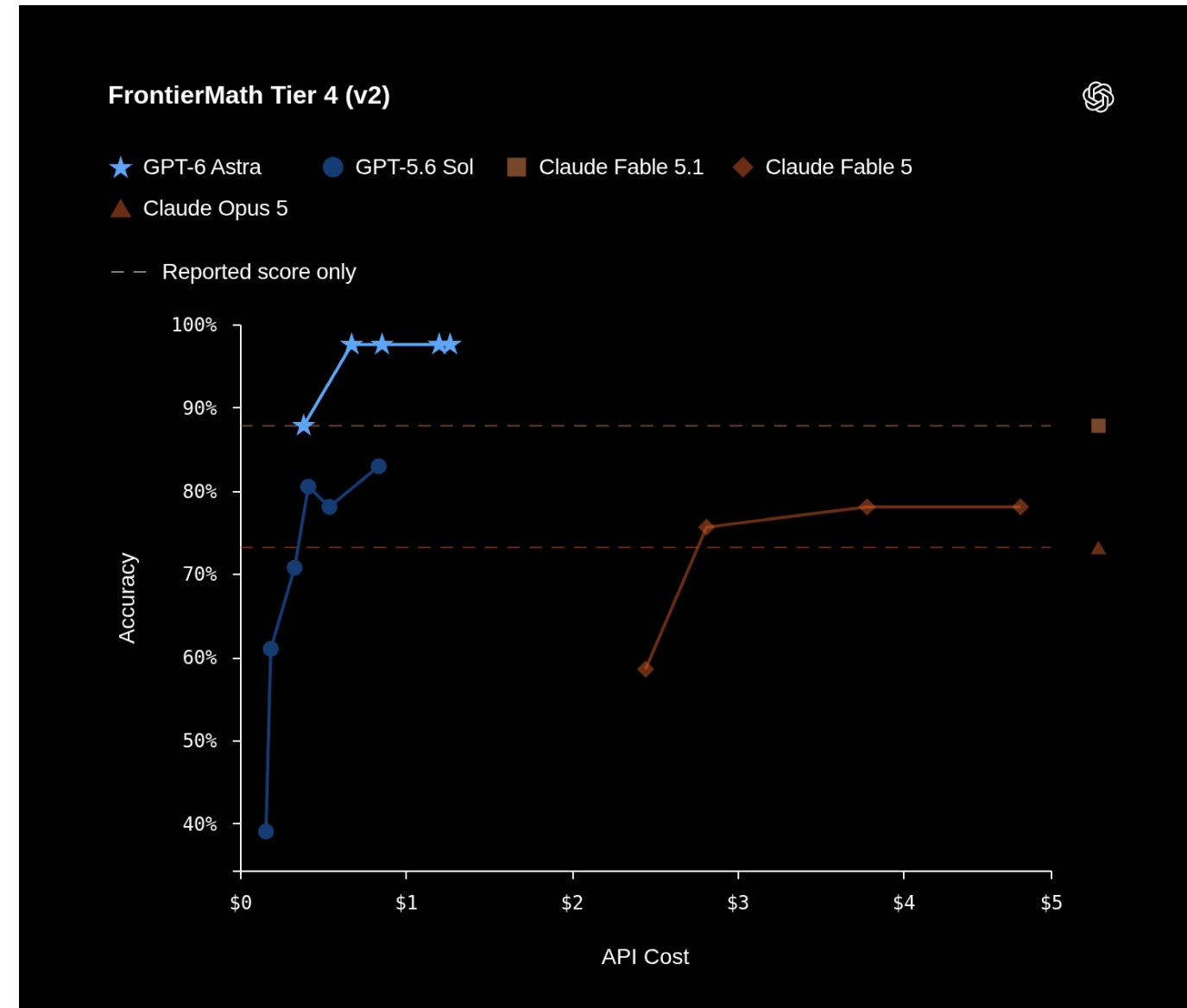


Claude Code quickstart ↗



Terminal-Bench Science 0.1 tests whether agents can complete scientific research workflows using code and terminal tools, including analyzing data, running simulations, and fitting models.

Source: [OpenAI, GPT-6 Astra launch results](#)



FrontierMath Tier 4 tests advanced mathematical reasoning on exceptionally difficult problems.

Source: [OpenAI](#), GPT-6 Astra launch results

Things I do not have time to talk about

- How to deal with data privacy?
- How to create skills?
- How to set up remote control and launch agents on a server/cluster?
- How to document the use of AI in research?
- What are the costs of using agents in research?
- How to stay up to date with all this?

Questions?