AI as a Research Assistant

Give your agent better tools

Archis Joglekar Phil Travis Jack Coughlin Manhar Gupta Jonathan Brodrick

WPI Plasma Kinetics — August 2026

Ergodic LLC

We look at the two-stream instability the way Ewart et al. do

Vlasov–Poisson, 1D–1V, normalized:

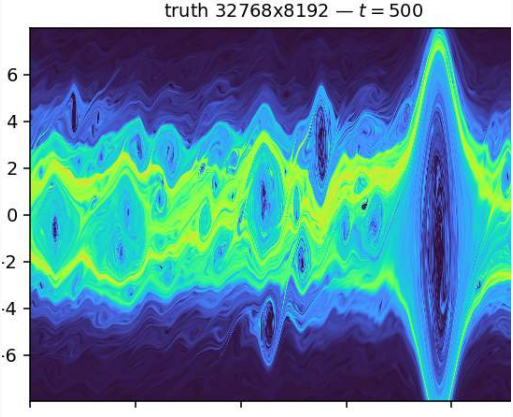
$$\partial_t f + v\,\partial_x f - E\,\partial_v f = 0$$

$$\partial_x E = 1 - \int f \,\mathrm{d}v$$

Two counter-streaming beams, population ratio r. Free energy goes into **filamentation** — ever finer scales in v, down to whatever your grid stops resolving.

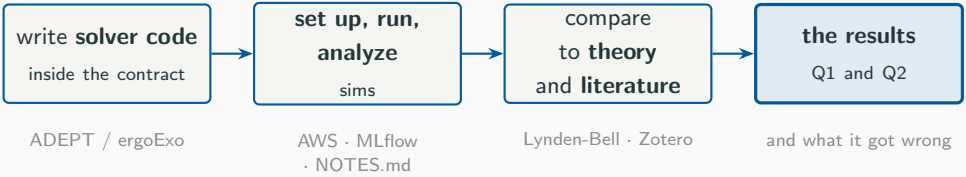
Q1. What does the energy distribution look like — does it reproduce Ewart *et al.*?

Q2. What if you cannot afford to resolve the cascade? Can you learn an operator to mimic it?



32768×8192, $t = 500\,\omega_p^{-1}$

We hand the whole problem to Claude Code



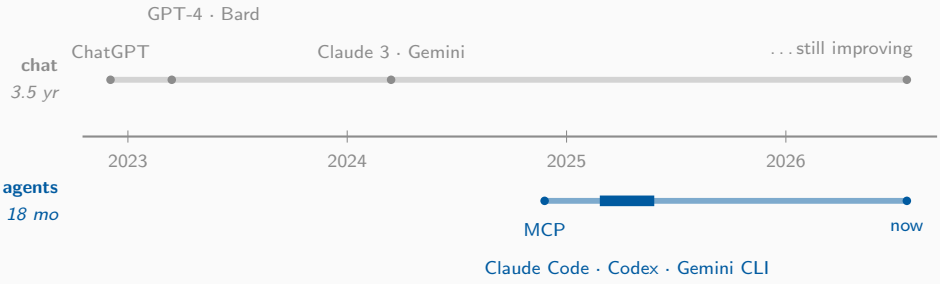
The rest of the talk is these four stages, and the tools each one needs.

Demo | before we start

```
> Launch a beam-ratio scan on AWS Batch --- 50/50, 60/40, 70/30, 80/20 --- at
a resolution that finishes inside this talk.  Four jobs, one per ratio.
```

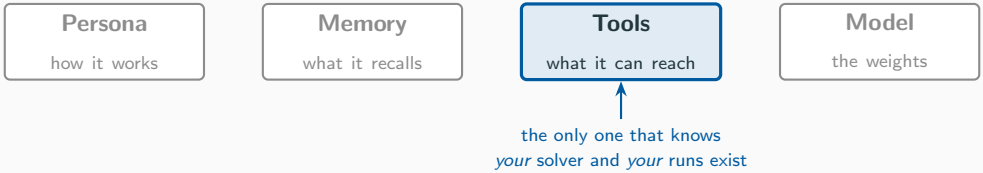
terminal: four configs written, code packaged and uploaded, four jobs submitted

2023 → now



An agent is the same model in a loop: *it can run a tool call, read the result, repeat.*

What do we mean by “agent”?



Persona, memory and the model are broadly the same for everyone in this room.

Tools are what specialize an agent toward *your* research

Stage 1 — write solver code inside the contract

ADEPT: one scaffolding, ten solvers

vlasov-1d
vlasov-2d
pic-1d
tf-1d
vfp-1d
envelope-2d
. . . 10 total

ergoExo — written once

mlflow run · config + units · param + artifact logging · val_and_grad

ADEPTModule — the checklist each solver fills in

write_units()	init_diffeqsolve()
get_solver_quantities()	get_derived_quantities()
init_state_and_args()	get_save_func()
init_modules()	post_process()
__call__()	vg()

helps decouple the numerical solvers from the experiment management and facilitates the addition of new solvers

Let the agent write code around anchor points, not the entire machinery

no contract

the whole solver

order of operations,
IO, logging,
physics — all of it

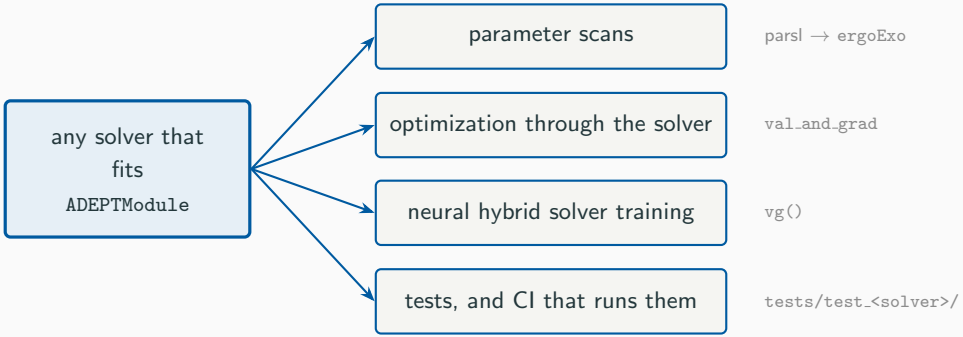
with a contract

the agent writes only these

fixed, and given

Its context holds one slot at a time rather than the whole object — and so does my review.

What you get once a solver fits the contract



None of this machinery is solver-specific, so the agent never rebuilds it. The same scan script runs on a laptop or a Perlmutter node unchanged.

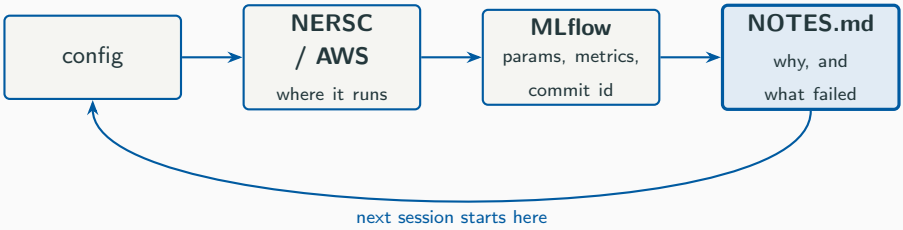
Demo | scaffold a new solver

```
> What does a new ADEPT solver actually have to implement?  Read ADEPTModule,
tell me which slots have workable defaults and which I genuinely have to
write, then stub out the full set for a new 1D solver.
```

terminal: the required set, then a stub with every slot present and empty

Stage 2 — set up and run

The loop: compute, record, remember



MLflow holds *what happened* — queryable months later, by description rather than run ID.

`NOTES.md` holds *why* — dead ends, negative results, why a parameter has that value.

Two clusters, one set of verbs

NERSC — allowlisted wrappers, one safe `ssh` each, so they are approved once instead of per call:

```
sync-up.sh
submit-batch.sh path
squeue.sh   sacct.sh   scancel.sh
read-log.sh   grep-log.sh
interactive-gpu.sh [hrs] [nodes]

remote-sha.sh
```

AWS — one generic job definition, the code shipped as a tarball:

```
aws batch submit-job \
--job-queue gpu \
--job-definition sim-gpu:2 \
--parameters \
code_uri=s3://.../$SHA.tar.gz,\
cmd="python run_two_stream.py
--cfg configs/r70-30.yaml",\
extras=gpu
```

The agent never writes a new launch path. It picks a verb, or fills in `cmd`.

Same loop either side: ship the code, start the job, watch it, pull the result.

Stage 3 — analyze, and compare against theory

Analysis is bespoke on purpose

Scaffolded

reused

The solver: ten slots, ten solvers, one workflow. Worth an interface because everything downstream depends on it.

Bespoke

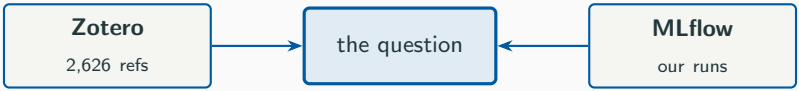
asked once

Every analysis question gets its own script. No contract, no interface, no reuse expected.

`analyze_lb.py`	197
`analyze_vortex.py`	157
`analyze_vortex_speed.py`	130
`debug_*.py` (4 files)	522
19 one-off scripts	*3,061*

One question each. Most were run once and never again — the four `debug_*` scripts existed to chase a single NaN.

Your paper library as a tool



"what exactly is the universal prediction,
and which diagnostic tests it against our runs?"

Pasting in a PDF works well. What the API adds is a question that spans the literature *and* our own runs.

2,626	references
26	collections
180	duplicates merged
280	open-access PDFs fetched

Demo | a question that needs both

```
> Find Ewart's papers in my library.  What exactly is the universal
prediction, and which diagnostic do I have to compute to test it against
our runs?
```

terminal: the library queried, then the answer — $N(\varepsilon)$, not $\langle f \rangle(v)$

What Ewart et al. predict

- no collisions ⇒ no route to a Maxwellian
- phase-space volume is conserved, and cannot be stacked
- mixing entropy ⇒ Fermi–Dirac, not Boltzmann
- turbulence drives it to a **universal** form

$$N(\varepsilon) \propto \varepsilon^{-2}$$

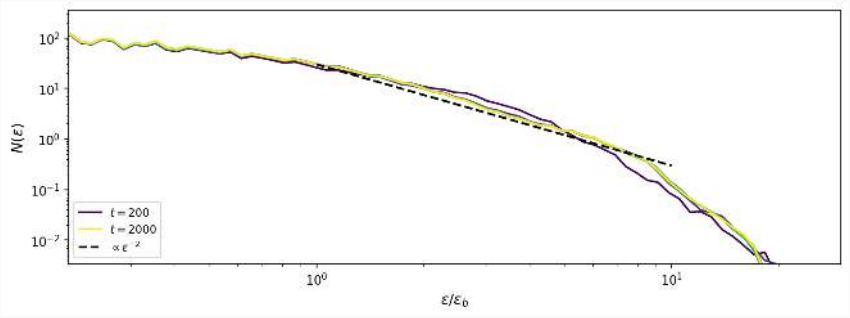
Universal — independent of how the beams were set up.

Ewart *et al.*, *PNAS* **122** (2025).

Can we reproduce -2, and does it survive unequal beams?

At 50/50 we reproduce the universal exponent

$N(\varepsilon)$ at eight times from $t = 200$ to 2000, saved by the run itself.



Slope over $\varepsilon/\varepsilon_b \in [1, 10]$: -2.17, steady from $t = 400$ to 2000.

A Maxwellian is exponential in ε — it curves away and never holds a slope.

Continuum Vlasov against their PIC — same exponent.

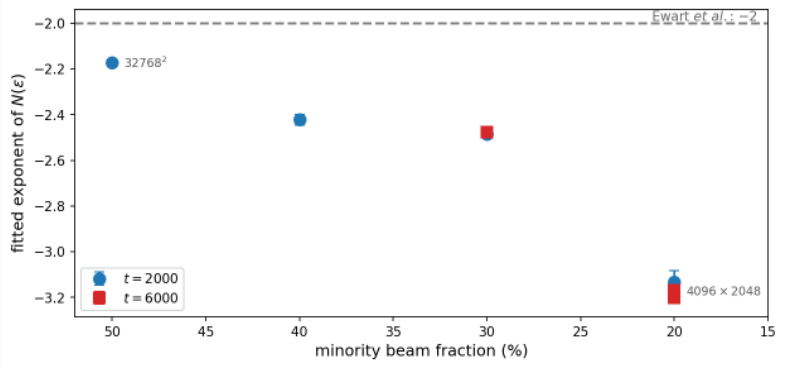
Demo | the scan we launched, finished

> The scan I launched at the start should be done. Fit the high-energy exponent of $N(\varepsilon)$ for each ratio, plot exponent against ratio, and tell me whether Ewart's -2 survives asymmetry. Then write the conclusion into NOTES.md.

exponent vs. ratio, four points — then the NOTES.md diff

Stage 4 — the results

The universal exponent survives only for equal beams



Steepens monotonically with asymmetry: -2.17, -2.42, -2.48, -3.17. Universality is recovered at 50/50 and lost as the beams become unequal.

Converged both ways — 70/30 gives -2.48 at $t = 2000$ and $t = 6000$; 80/20 gives -3.13 at 2048^2 and -3.17 at 4096×2048.

Caveat: ε is lab-frame and the unequal runs carry a net drift — the trend is robust, the individual exponents are not final until this is redone in the momentum frame.

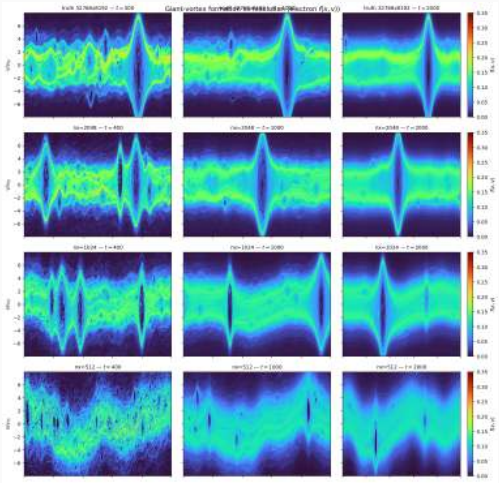
Q2. What if you cannot resolve it?

Filter onto a grid with $\Delta x \gg \lambda_D$. Every term commutes with the filter except the nonlinear one:

$$\partial_t \langle f \rangle + v\, \partial_x \langle f \rangle - \langle E \rangle\, \partial_v \langle f \rangle \;=\; \underbrace{\langle E'\, \partial_v f' \rangle}_{\text{unclosed}}$$

That term is the whole problem. First: what happens if you simply drop it?

Keeping the vortex alive takes 2–4 points per Debye length



$f(x, v)$; lower three rungs omitted (no structure at any time).

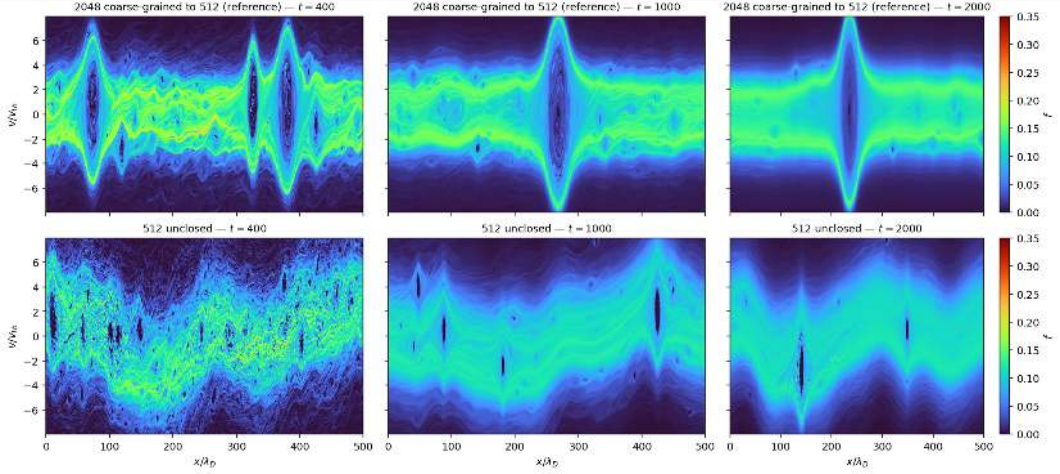
$n_x \geq 2048$	one persistent vortex
$n_x = 1024$	coherent, but two
$n_x = 512$	remnant slivers only
$n_x \leq 256$	smeared bands

At $n_x = 512$ the Debye scale is resolved and 98.9% of γ^2 captured, and the vortex is still lost.

Retaining it for $2000\,\omega_p^{-1}$ needs 2–4 points per Debye length.

Bulk convergence and structure retention are different thresholds.

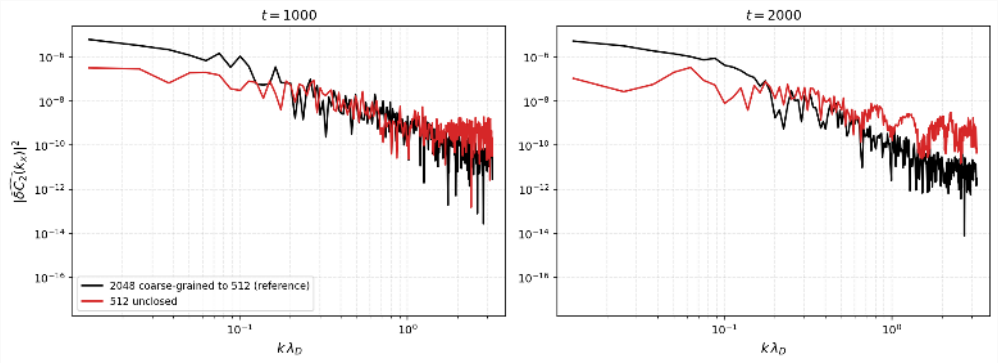
Dropping it: the coarse run loses the structure



Top: the 2048 run coarse-grained onto 512 — what a perfect 512 field would look like.

Same grid, same physics, no sub-grid term — and the vortices are gone by $t = 2000$.

And in the spectrum: the small scales are not the problem



Unclosed (red) against the coarse-grained **reference** (black): across the resolved band the amplitudes sit on top of each other.

It does not fail by having the wrong small scales — it fails by **under**-dissipating them. Nothing removes the flux.

That is what the closure has to supply: a sink, not structure.

What we want: a drag–diffusion operator we do not have to write

Stand in for the missing term with conservative drift-diffusion in v:

$$\partial_v[D\,\partial_v\langle f\rangle + C\,\langle f\rangle]$$

$$D = \frac{\alpha_1(\partial_x E)^2}{v^2+\alpha_2^2} + \alpha_3, \quad C = \beta_1(\partial_x E)^2 v$$

Four scalars, quasilinear form after Nastac *et al.* (2025), trained through the differentiable solve.

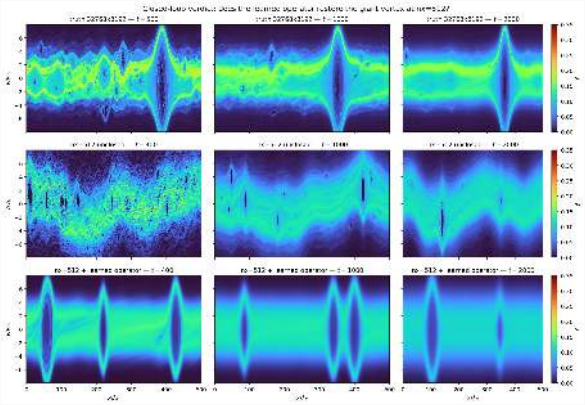
And the operator already existed

ADEPT ships a generic conservative drift-diffusion scheme — flux form, zero-flux boundaries, density conserved exactly — shared with Lenard-Bernstein and Dougherty.

Only D and C had to be written: 115 lines. The agent could not break conservation, because conservation belongs to the scheme it calls.

Target: not the cascade itself — unrepresentable at $\Delta x > \lambda_D$ — but its *flux*, the rate $C_{2,0} = \frac{1}{2}\int f_0^2\,\mathrm{d}v$ is destroyed.

The operator form restores the vortex; training only tunes it



Trained operator restores vortex persistence at $n_x = 512$; $\langle e^2 \rangle$ drops from a 10–20× pile-up to roughly truth level.

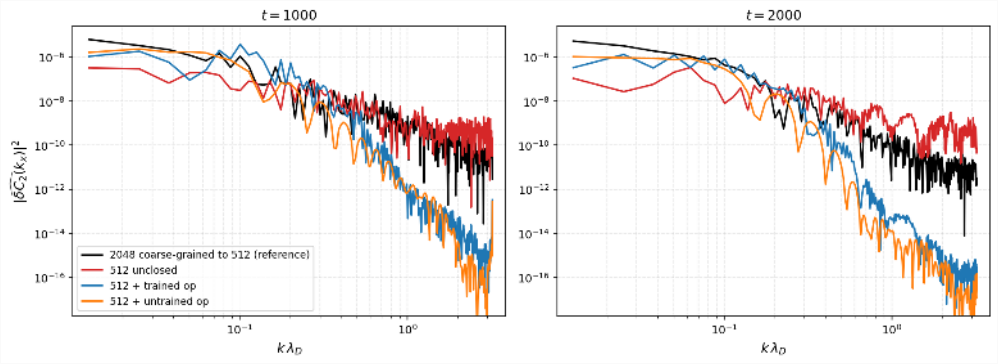
The untrained control — same form, default coefficients — also restores it.

log f_0 err	500	1000	2000
unclosed	1.249	0.534	0.539
+ untrained	0.665	0.367	0.274
+ trained	0.455	0.342	0.235

Form buys the coherence; training tunes amplitude.

It over-smooths, so the next operator must be scale-selective

Same axes — now the **closed** runs.



Closed: loses 3–6 decades above $k\lambda_D \approx 0.3$.

Unclosed: **under**-dissipates at small scales ($\langle e^2 \rangle$ up 10–20×), **over**-dissipates at large.

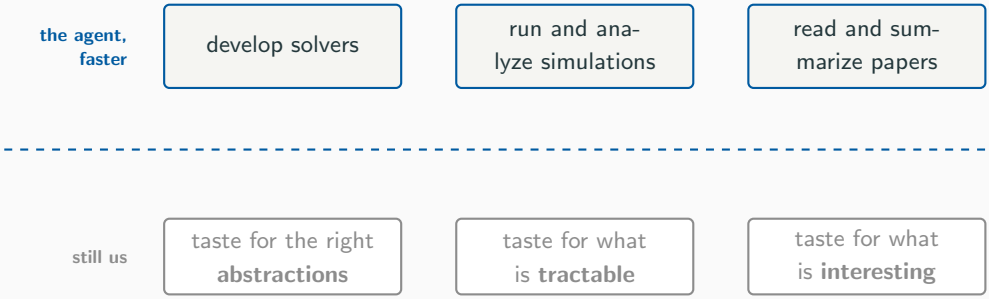
So the correction must change sign too — damp only near the grid cutoff.

What stayed human

What does a computational physicist do?

short term	develop solvers	run and analyze simulations	read and summarize papers
long term	taste for the right **abstractions**	taste for what is **tractable**	taste for what is **interesting**
in practice	config, units and logging belong in the runner — not in each of 10 solvers	growth rate converges at $n_x{=}512$; the vortex needs $n_x{\geq}2048$ — 2 h/run	unequal beams carry two phase-space levels — outside the published single-level case

Where the line falls



Everything above the line, it does faster than I do

Building the tools

All of it is 1,800 lines of markdown and shell

`skills/nersc-workflow/` sync, launch, monitor, pull, cancel	307
`skills/adept-run/` one run vs. a parameter scan	294
`skills/mlflow-query/` experiments, runs, metrics, artifacts	130
`scripts/ops/` squeue, scancel, sync-up, read-log, mlflow-get-params, submit-batch, ...	1,057

`examples/first-run/`

30-second job, exercises the whole loop

```
git clone .../ergodicio/ergodic-claude
./scripts/bootstrap-local.sh
```

A skill is just a file

```
---
name:  mlflow-query
description:  Query the tracking
server for runs, metrics,
artifacts.  Use when checking
training progress.
---

## Setup
The server sits behind a proxy
exposing /ajax-api/ instead of
/api/.  A bare 'import mlflow'
404s on every call.  Use:
import _mlflow_patched as mlflow
```

That last paragraph is the whole point. A skill is mostly somewhere to write down the thing that would otherwise cost an hour every time.

Three failures, and what actually caught them

- **Right code, wrong array.** I first built $N(\varepsilon)$ from the *downsampled* f. Every run appeared to freeze after $t \approx 300$, and there was a tidy story for it — the giant vortex shutting the cascade off. But $1/(k\Delta v) \approx 297$ on that grid: it is where the *downsample* stops resolving filaments. Caught by recognizing a timescale, not by a test.
- **A job in `RUNNING` need not be running.** NaNs, a skip loop, and a job healthy by every state we were watching — for a day. Monitors now demand a metric heartbeat, not a job state.
- **One narrow verb per script.** `cancel-job <id>` is approved once and reused forever; `ssh <host> "$CMD"` can never be blanket-approved, so it asks every time — and by the twentieth prompt you are clicking, not reading. The failure is approval fatigue, not `ssh`.

All three passed every check that was running at the time — and none of the fixes are AI work. It is the experiment hygiene we had been meaning to do for years.

Summary and open questions

What I claim

- An agent is a model in a loop. It already knows the physics; **tools** are how it reaches *your* solver and *your* runs.
- **Scaffold** what gets reused and fails quietly — ten solvers, one workflow, the agent filling named slots.
- Leave **bespoke** what is asked once and fails loudly — 3,000 lines of throwaway analysis.
- The **record is the memory**: MLflow for what happened, `NOTES.md` for why.

Open questions

- Does Lynden-Bell survive beam asymmetry, or have those runs simply not finished relaxing?
- Is "coherent structure persists" the right pass/fail criterion for a closure?
- Could a theorist actually use `sympy` this way? I have not tried it and I do not know.
- Where would you refuse to let this near your work?

These slides are beamer, and Claude Code wrote most of the LaTeX — including every diagram in this deck.

Questions.