AI for PDE

GPT + Codex
from ideas to auditable research

Work / Dots / Rethlas / Danus / Pharos / DeepSeek API

THINK

Scope and conjectures

COMPUTE

Files and experiments

VERIFY

Independent evidence

From a question to a research workflow

A workshop with a worked Vlasov research case and an installation appendix.

01

Choose the layer

Separate a model, a tool interface and a research orchestrator.

02

Specify a claim

Freeze assumptions, quantifiers, constants and evidence requirements.

03

Run a local gate

Use Rethlas for a focused lemma and Danus for dependencies.

04

Deploy safely

Check authentication, APIs, isolation, logs and explicit budgets.

The objective is a reviewable result, not the longest answer or the largest worker swarm.

Three layers, three different decisions

Changing the orchestrator does not necessarily change the underlying model.

01 / INTELLIGENCE

Model

GPT or DeepSeek supplies reasoning and tool-call outputs. Choose an available model and supported settings.

02 / EXECUTION

Interface

ChatGPT, Work, Dots and Codex connect models to conversation, files, tools and execution.

03 / ORGANIZATION

Orchestration

Rethlas, Danus and Pharos organize proof search, review cycles and research state.

Discuss with GPT. Execute with Codex.

A recommended division of labor, not a hard boundary between capabilities.

Clarify

Equation and quantifiers

›

Explore

Models and counterexamples

›

Execute

Code and proof files

›

Review

Constants and dependencies

Human judgment	Choose the question, distinguish evidence from proof, and authorize scope changes.
Persistent artifacts	A frozen statement, reproducible commands and an exact report connect successive rounds.

Write a PDE task contract

Freeze the problem before allowing the agent to explore.

Object	Equation, dimension, domain, boundary conditions and solution concept.
Quantifiers	All data or iid data? Fixed T or global time? Expectation or high probability?
Assumptions	Regularity, weights, moments, smallness and already established lemmas.
Constants	State dependence on N, T, cutoffs, dyadic scales and the initial density.
Acceptance	A complete local proof, a valid counterexample, or the first exact gap.

No silent strengthening of hypotheses and no promotion of numerical tests into theorems.

A reusable GPT proof-audit prompt

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

```
Read PROBLEM.md and the full candidate proof.
Restate the target and every allowed assumption.
Check each load-bearing estimate independently.
Inspect endpoints, sums, limits and constant dependence.
Identify the earliest unproved implication.
Do not silently supply an extra lemma.
Return valid steps, the first gap and a minimal repair.
Version any statement that needs new assumptions.
```

prompts/GPT_AUDIT.md

01 **Read the argument**

A success summary is not a proof.

02 **Find the first gap**

Do not spend effort polishing steps that already depend on it.

03 **Repair locally**

A weaker result must be labeled as a different claim.

Install Codex and begin read-only

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
npm install -g @openai/codex
codex --version
mkdir -p ~/pde-lab
cd ~/pde-lab
git init
codex

# First task: inspect files and assumptions only.
```

Review paths, permissions and model settings before execution.

01

Prerequisites

Check Node.js, npm, Git and Python first.

02

Authentication

Use the account or API route actually configured for this environment.

03

First session

Confirm model and permissions before authorizing edits.

Store research state in the project

Keep framework contracts intact; add project-specific instructions separately.

```
pde-lab/
  AGENTS.md
  PROBLEM.md
  sources/
  candidates/
  reviews/
  experiments/
  logs/
  STATE.md
```

Read-only sources

Preserve the original paper and its version.

Exclusive outputs

Give each worker a separate writable directory.

Explicit dependencies

Keep the exact supporting statements and review status.

Framework boundary

Never hand-edit Danus fact_graph or overwrite its AGENTS.md.

Capture three kinds of evidence

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
mkdir -p logs
codex exec --sandbox read-only --json \
  -o logs/audit.md \
  "Audit PROBLEM.md; list missing definitions." \
  > logs/events.jsonl 2> logs/stderr.log

codex --version
git rev-parse HEAD
```

Review paths, permissions and model settings before execution.

01

Mathematical output

The report should identify statements, assumptions and gaps.

02

Execution record

JSONL and stderr show calls, completion and failures.

03

Environment

Record commits and actual routing; never record secrets.

Work is organized around a deliverable

Here, Work means ChatGPT Work, not a generic worker process.

01

Research brief

Compare theorem assumptions from supplied sources and produce a cited report.

02

Presentation

Turn reviewed results into a deck, handout or progress summary.

03

Tool-assisted analysis

Use approved files and tools, with an explicit result to inspect.

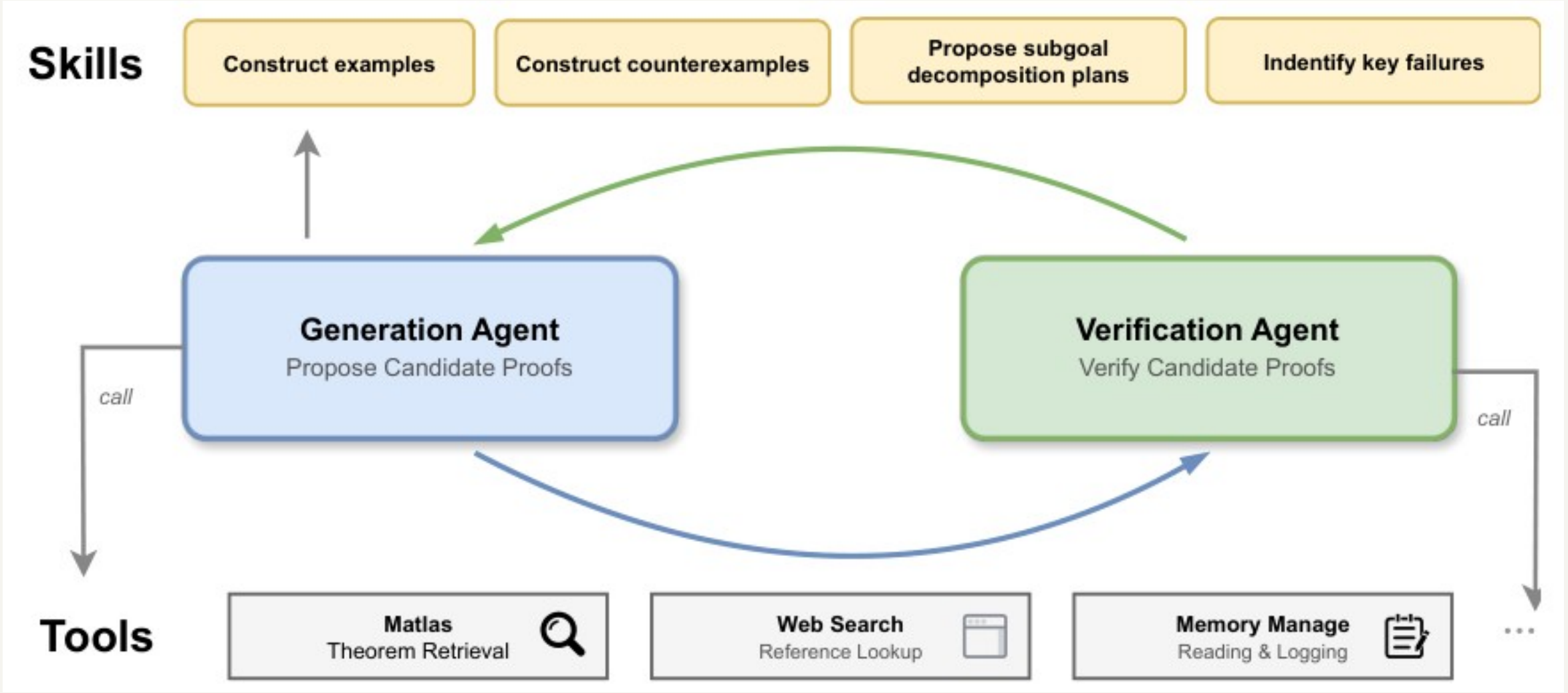
Work and Codex can both use subagents; available tools and environments depend on the account.

Work, Rethlas and Danus: compare structure

These are workflow recommendations, not benchmark rankings.

Dimension	Work	Rethlas	Danus
Unit of work	Reviewable deliverable	Problem / local proof	Project / claim graph
Core pattern	Tools and subagents	Generate, verify, repair	Main, workers, verifier
Persistent state	Files and sessions	Drafts, memory, reviews	Memory and fact graph
Suggested use	Synthesis and artifacts	A fragile local lemma	Dependent proof modules
Extra overhead	Tool permissions	Local verifier service	Services and scheduling

Rethlas: the original architecture



Original Figure 2 / Automated Conjecture Resolution with Formal Verification, arXiv:2604.03789v2

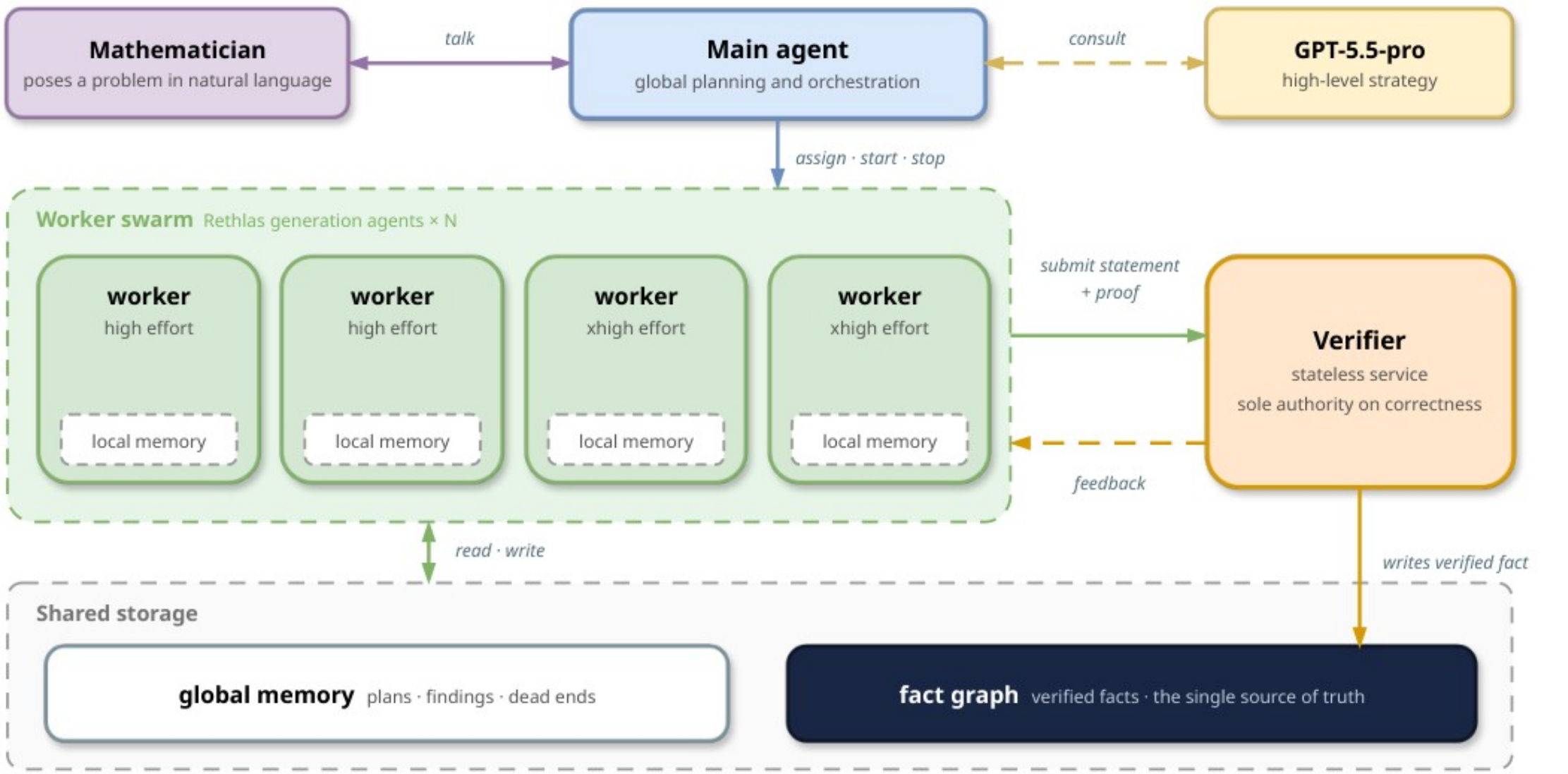
Read the Rethlas loop, step by step

One focused claim remains the unit of review, even when subgoals are delegated.

1 Generate	Read the exact statement; construct a candidate proof or a testable counterexample.
2 Use skills	Try examples, decompose the goal, retrieve applicable results and isolate failed steps.
3 Verify	Send the statement and complete proof to a separate verification service.
4 Repair	Use the raw rejection to fix the first error; preserve the original hypotheses.
5 Export	Save the proof, constants, memory and verdict; state the precise review level.

For Vlasov: start with CT-FREE-01, not the entire endpoint propagation-of-chaos theorem.

Danus: the original architecture



Original Figure 1, arXiv:2607.06447v2. Model labels are historical; the codex branch has evolved.

Read Danus as a dependency-aware research team

The fact graph is a workflow ledger, not a formal proof kernel.

01 / COORDINATION

Main

Freezes scope, assigns tasks and replans when the evidence changes.

02 / PROOF SEARCH

Workers

Explore one claim at a time and submit a full proof with predecessor IDs.

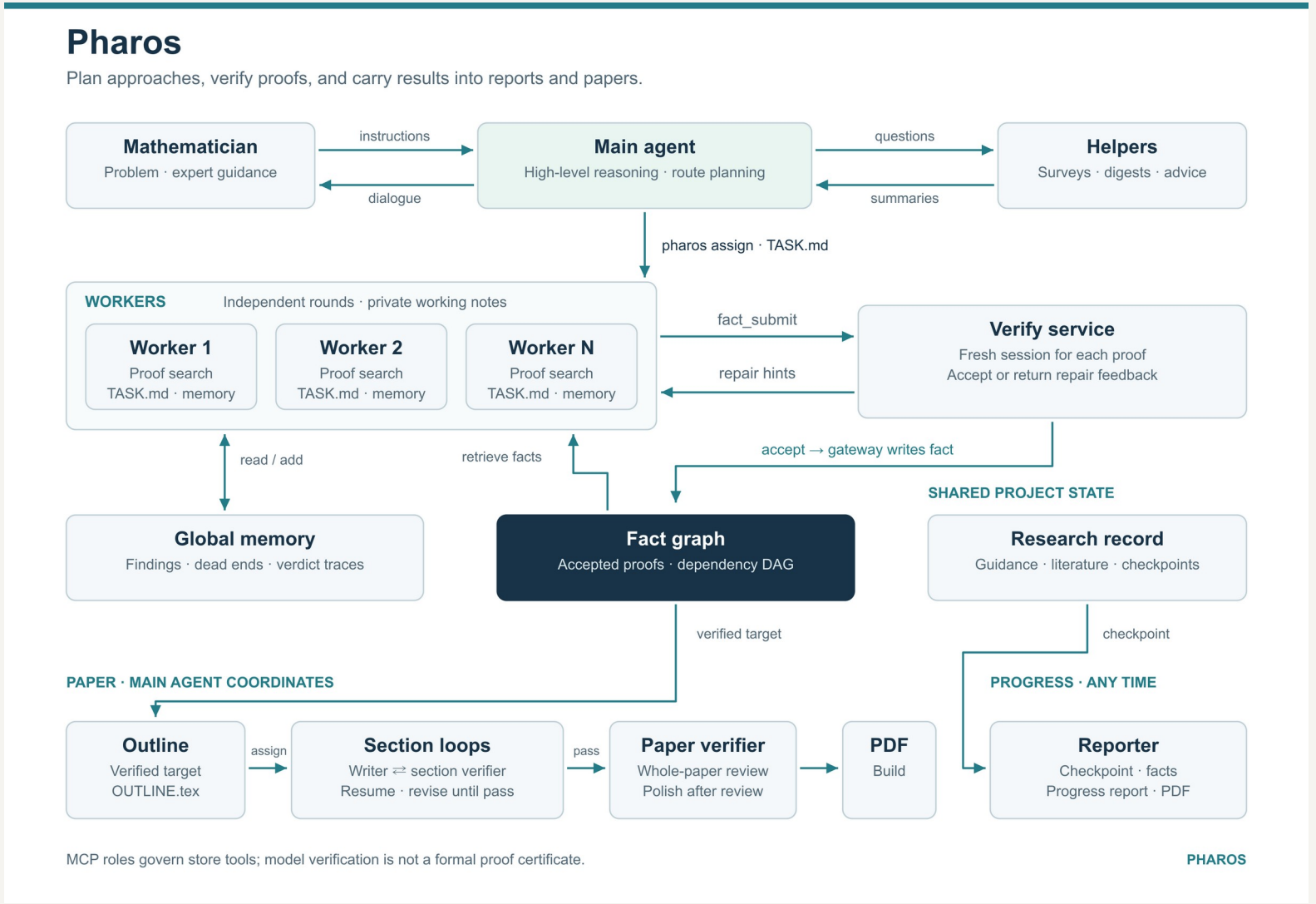
03 / REVIEW

Verifier + gate

A fresh verifier returns a verdict; the submission gate admits accepted facts.

Global memory	Plans, failed routes and findings; useful context, not automatically a proved premise.
Fact graph	Versioned statements and dependencies; a revoked claim triggers downstream review.

Pharos architecture



[Original diagram from the FrenzyMath Pharos architecture guide]

A first Pharos research run

Use a route board to compare competing approaches to an open-ended problem.

COPYABLE PROMPT FOR ROOT OPS

Prepare a VLASOV-A1 pilot using the supplied brief and papers. Show me PROBLEM.md and your initial guidance for approval before launching.

Start with CT-FREE-01. Compare a direct pair estimate with conditioning on the shared particle. Keep the Vlasov endpoint as a research target.

Record the assumptions, evidence and first obstacle for each route. Show me what needs review next.

01 **Set up with root Ops**

Open Codex in the Pharos repo. Give Ops the problem, material paths and scope.

02 **Steer the project Main**

Review ROUTES.md and redirect stalled routes. Workers submit proofs to an independent verifier.

03 **Read the progress report**

Run pharos report <project>. Check the cited facts before reusing a claim.

ChatGPT Dots for research coordination

Give your dot an ongoing responsibility with clear limits and review points.

COPYABLE PROMPT FOR YOUR DOT

Coordinate a pilot for CT-FREE-01 using the problem brief and the supplied references.

In parallel, check the cited hypotheses, ask Codex to run a small free-transport diagnostic, and have a fresh reviewer challenge the first variance bound.

Bring back source links, the script and run log, and the first unresolved step. Pause for my review before expanding the claim.

01 **Set up your dot**

Open Dots in ChatGPT and follow setup. Share the brief and connect any required apps.

02 **Delegate and follow up**

Run checks in parallel. Local Codex needs a connected computer with ChatGPT open.

03 **Review in Activity**

Open the dot's profile, then Activity. Inspect task files, results and decisions that need you.

Choose by task shape, not by model prestige

Match each tool to the evidence you need.

A report or deck	Use Work with sources, page requirements and a reviewable deliverable.
An uncertain lemma	Use Rethlas for exact calculations, counterexamples and local repair.
A dependent project	Use Danus to separate modules, track failed routes and control assembly.
A numerical check	Use Codex for code, parameters, reproducible runs and saved diagnostics.

Use Pharos to explore competing routes and Dots to coordinate ongoing follow-up.

What does “verified” actually mean?

Keep the evidence level visible in every handoff.

Candidate	A plausible argument with no independent check.
Computational check	A script or finite test passes within its stated coverage.
LLM-reviewed	An independent model review passes; shared omissions remain possible.
Human-reviewed	A specialist checks the statement, proof and supporting dependencies.
Formally checked	A trusted proof kernel checks a formal statement whose meaning is also audited.

A hash proves file identity. A “correct” label records a review. Neither alone proves a theorem.

Different workers should produce different evidence

Start with a small team and reserve an independent review step.

A

Structure

Scaling, identities, symmetries and exactly solvable models.

B

Estimates

Norms, endpoints, constants, probabilities and scale sums.

C

Adversary

Counterexamples, extreme regimes and circular assumptions.

V

Reviewer

Fresh-context reconstruction of the frozen candidate.

Main coordinates the evidence; mathematics is not decided by majority vote.

A general Main-agent instruction

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Freeze one local target and its admissible inputs.
Delegate structure, estimates and adversarial tests.
Give each worker an exclusive writable directory.
Require statements, proofs, constants and dependencies.
Freeze candidate files before independent review.
Keep conditional results and unresolved claims visible.
After two unchanged failures, request a local replan.
Respect the authorized concurrency and budget.

01 **Bound the task**

A local claim is easier to audit than a complete research program.

02 **Separate evidence**

Parallel work must not create circular premises.

03 **Control execution**

Runtime limits are separate from prompt instructions.

Worker and verifier receive different contracts

A fresh reviewer should see the argument, not the author's confidence.

WORKER

State the exact claim.

Derive every load-bearing estimate.

Name the source and predecessor.

Keep all scale and cutoff losses.

A valid counterexample is a result.

VERIFIER

Recompute the key implication.

Check assumptions and quantifiers.

Inspect sums, limits and tails.

Locate the first error or missing input.

Do not silently repair the proof.

Parallelize independent work; gate dependent work

A conditional lemma must retain its hypotheses when it enters a downstream argument.

Freeze
Definitions and scope

›

Explore
Independent modules

›

Review
Exact local claims

›

Assemble
Only supported inputs

Before review	Sibling candidates may be discussed as conjectures, not cited as established facts.
After rejection	Keep the failed statement and revise or invalidate every affected dependent claim.

A graph organizes claims; it cannot compensate for a false edge.

Native Codex subagents: use the installed interface

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Delegate three read-only investigations:
1. Audit the frozen assumptions.
2. Derive the fixed-scale model estimate.
3. Search for a counterexample.
Wait for the three reports before starting review.
Do not let agents edit the same proof file.

Use /agent to inspect their threads.

01 **Check availability**

Current releases expose subagents, but account and client settings matter.

02 **Use real settings**

Inspect the current configuration reference; do not copy obsolete keys.

03 **Respect frameworks**

Do not overwrite Rethlas or Danus agent configuration.

Monitor mathematics and infrastructure separately

A running process is not evidence that the proof is advancing.

01

Mathematical delta

A new bound, a valid obstruction, a smaller loss or a newly closed dependency.

02

Execution health

Successful requests, correct routing, changing artifacts and bounded retries.

03

Replanning signal

A disproved premise or two unchanged failures should trigger a narrower next test.

Keep a live supervisor: the Danus operating guide does not promise strategy replanning when Main is inactive.

Worked case: the 3D Vlasov α = 1 endpoint

A candidate high-probability rate, not an already completed proof.

$$\mathbb{P}\left[\sup_{0\le t\le T} W_1\left(\mu_N(t), f(t)\right) > C_T N^{-1/6} (\log N)^A\right] \le C_T N^{-c}$$

SCOPE

Dimension

Physical space: 3.
Phase space: 6.
Force size: $|K(x)| \lesssim |x|^{-1}$.

QUANTIFIERS

Probability

lid initial data.
Fixed time horizon T.
Derive A, c and all constants.

CANDIDATE

Status

Reference estimates and actual-flow transfer are different gates.

This is not 3D Coulomb. Uniform-in-time marginal W_1 is not automatically a path-space theorem.

Freeze the particle and PDE models

The pilot kernel is illustrative; changing the kernel class requires a new contract version.

$$\dot{X}_i = V_i, \qquad \dot{V}_i = \frac{1}{N}\sum_{j\neq i} K(X_i - X_j), \qquad |K(x)| \lesssim |x|^{-1}$$

$$\partial_t f + v\cdot\nabla_x f + (K * \rho_f)\cdot\nabla_v f = 0, \qquad \rho_f = \int f\,dv$$

Initial law	$f_0 \in C^2_c(R^6)$, a probability density; Z_i^0 are iid.
Reference solution	Fix a classical Vlasov solution with explicit support and regularity bounds on [0,T].
Uncut flow	Collision handling, particle well-posedness and regulator removal remain separate obligations.

Representative $K(x) = x/|x|^2$; smooth pilot $K\eta(x) = x/(|x|^2 + \eta^2)$. No uniform-in-$\eta$ claim is assumed.

One reduction, two sources of error

Use independent nonlinear reference particles driven by the fixed Vlasov solution.

$$W_1(\mu_N(t), f(t)) \le \frac{1}{N} \sum_{i=1}^{N} |Z_i(t) - \bar{Z}_i(t)| + W_1(\bar{\mu}_N(t), f(t))$$

STANDARD MODULE

Sampling error

The reference empirical measure is the pushforward of an iid sample. Lipschitz transport reduces this term to time zero.

NEW RESEARCH

Dynamical error

The interacting particles are correlated. Singular-force stability and tail control must be proved without assuming closeness.

The same-label coupling gives an upper bound for W_1; it does not create a stability estimate.

Four gaps that a good prompt must preserve

Do not let an attractive outline silently become a theorem.

01

Truncated cost → W_1

A bound on average $\min(e_i,1)$ needs a quantitative tail estimate.

02

Static A → adaptive A(t)

Selection depends on the paths; a fixed-subset bound does not automatically transfer.

03

Reference tube → real force

A larger event does not control an inverse distance at a much smaller real separation.

04

Fixed scale → all scales

Record actual R^{-1}/R^{-2} weights, microscopic cutoffs and union-bound costs.

The frozen Vlasov problem file

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

```
Project: VLASOV-A1. Status: RESEARCH TARGET.
Fix d_x = 3 and force exponent alpha = 1.
Name the kernel, initial law and reference-flow bounds.
State the exact high-probability W1 target.
Distinguish free, nonlinear-reference and real flows.
Authorize CT-FREE-01 as the first local task.
Do not assume separation, tail control or cutoff removal.
Version every change of assumptions or metric.
```

PROBLEM.md

01

Common vocabulary

All workers must refer to the same equation and flow.

02

Explicit frontier

The main theorem motivates the work but is not a premise.

03

Version control

A weaker claim is useful only when its altered scope is visible.

Two complementary runs, not two duplicate proofs

Use one shared problem contract and exchange frozen artifacts.

LOCAL SEARCH

Rethlas

Investigate the free-flow collision kernel. Derive geometry, mean and variance; test the first scale obstruction.

PROJECT COORDINATION

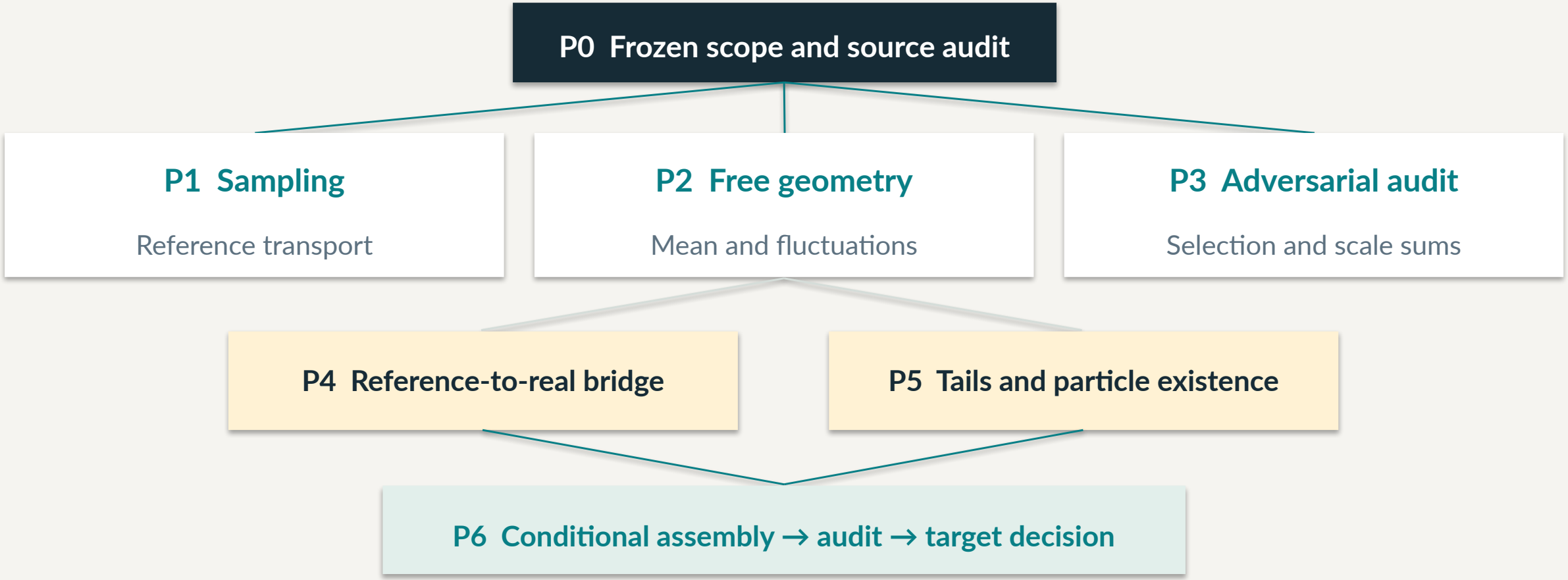
Danus

Audit the source, split the probability and transfer modules, review imports and manage the dependency graph.

Rethlas output enters Danus as a candidate, never as a pre-approved fact.

Vlasov dependency graph: where work may proceed

Six logical jobs can be scheduled in waves of at most three exploration workers.



Amber nodes are not assumed true. Review each dependency before changing a node's status.

Danus: six logical jobs and one independent reviewer

Map job labels to actual runtime worker IDs; they are not invented command names.

Job	Responsibility	Required deliverable
W0	Source and endpoint audit	Exact failure map in the original proof
W1	Free collision geometry	Residence identity; slow / fast regimes
W2	Mean, variance, force fluctuation	Parameter-resolved probability bounds
W3	Adaptive selection and scale audit	Valid uniformity class or counterexample
W4	Real-flow stability and regulator	A proved bridge or first missing integral
W5	W_1 tails, existence and assembly	Conditional closure with owned premises

V reviews frozen candidates in a fresh context. Main coordinates; no worker votes a theorem into existence.

Danus Main prompt ①: scope and delegation

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

```
Read the installed contracts and PROBLEM.md.
Create or reuse project vlasov-a1 without overwriting it.
Record commit, CLI version and actual routing.
Make CT-FREE-01 the first authorized target.
Map W0–W5 to real worker IDs from the runtime.
Start W0, W1 and W2; cap exploration at three workers.
Keep W4/W5 conditional until dependencies are reviewed.
Do not assume the final rate or particle separation.
```

prompts/DANUS_MAIN.md

01 **Freeze**

Prevents dimension, metric and hypothesis drift.

02 **Delegate**

Each job must own one kind of evidence.

03 **Gate**

Downstream workers may audit a gap, but may not assume it solved.

Danus Main prompt ②: review, repair and stop

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Give each worker its own writable workspace.
Require statement, proof, constants and predecessors.
Use the native submission/verifier gate for every fact.
Import Rethlas artifacts as unreviewed candidates.
Never hand-edit the fact graph or suppress a rejection.
Update FIRST_UNRESOLVED after every mathematical delta.
Two unchanged failures trigger a targeted replan.
Finalize only after human approval of the full target.

`prompts/DANUS_MAIN.md`

01 **Review boundary**

A candidate summary cannot become a premise.

02 **Honest repair**

New assumptions require a newly versioned claim.

03 **Stop condition**

Use runtime caps; a prompt is not a hard budget control.

W0: audit what the original theorem actually proves

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Read the exact Hauray–Jabin source version.
List every step that uses alpha < 1.
For each: quote its location, assumptions and loss.
Substitute alpha = 1 and classify the failure.
Separate logarithms, power losses and probability gaps.
Audit the metric, cutoff and particle-flow definition.
Check current literature before claiming novelty.
Return the smallest ordered list of genuinely new lemmas.

`prompts/W0_SOURCE_AUDIT.md`

01 **Input**

The original paper, not a remembered theorem.

02 **Work**

Identify which proof mechanisms truly need replacement.

03 **Acceptance**

A referenced obstruction ledger, not a claim that all losses are logs.

W1: compute the free-flow collision interval

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

For r(t) = r0 + t w, compute the exact interval.
Treat w = 0, grazing and empty intersections.
Prove length ≤ min(|I|, 2R/|w|) when w ≠ 0.
Separate |w| ≤ U from U < |w| ≤ 2U.
Use 2R/U only in the annular case.
Keep impact parameter and endpoints in the impulse.
Do not replace a curved characteristic by a line.
Return the identity, regime table and boundary tests.

`prompts/W1_GEOMETRY.md`

01

Input

Deterministic r0, w, R and time interval I.

02

Work

Remove geometric guesswork before doing probability.

03

Acceptance

An exact free-flow statement with all degeneracies treated.

W2: respect pair dependence and use error slack

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Prove the fixed-scale mean bound for Gamma_N.
Normalize the symmetric pair sum exactly.
Compute the one-point and degenerate projections.
Derive the full Hoeffding variance identity.
Do not treat pairs sharing an index as independent.
Bound the integrated reference-force error in expectation.
Use Markov at epsilon_N before demanding a stronger rate.
Keep every N, R, U, I and self-term contribution.

`prompts/W2_PROBABILITY.md`

01 **Input**

Iid reference particles and the exact pair kernel.

02 **Work**

Separate geometry, dependence and probability costs.

03 **Acceptance**

An explicit bound with a complete parameter ledger.

W2 example: do not over-strengthen the fluctuation target

At the final $N^{-1/6}$ scale, a first-moment estimate may already buy a polynomial failure bound.

$$\mathbb{E}\,F_N \le C_T N^{-1/2} \quad \Longrightarrow \quad \mathbb{P}(F_N > \varepsilon_N) \le C_T N^{-1/3} (\log N)^{-A}$$

Define F_N	The time integral of the mean absolute empirical force error along independent reference particles.
Use local L^2	For $\alpha = 1$ in R^3, $\lvert K\rvert^2$ is locally integrable. Conditional variance yields a candidate $N^{-1/2}$ bound.
Do not overclaim	This does not control the interacting force, nor the supremum of every particle's error.

A worker must prove the finite-moment and support assumptions; the algebraic Markov step is explicit.

W3: attack selection and dyadic summability

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Distinguish one fixed A, all static A, and moving A(t).
State the selection class and any filtration explicitly.
Do not hide a union bound over 2^N subsets.
Test small subsets, rare large rows and slow pairs.
Compute the actual R^{-1} and R^{-2} weighted sums.
Keep the lower scale, velocity layers and log losses.
A static-subset estimate does not justify time switching.
Return a lawful estimate or a concrete counterexample.

`prompts/W3_SELECTION_SCALES.md`

01 **Input**

The precise estimate proposed by geometry/probability.

02 **Work**

Try to break the extension actually needed in stability.

03 **Acceptance**

An owned uniformity class and a closed scale ledger.

W4: the reference-to-real bridge is a separate theorem

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Name every reviewed reference estimate you use.
Define the real flow and a measurable stopping time.
Audit tube inclusion separately from force control.
Test real distances much smaller than the path error.
Identify the missing microscopic impulse estimate.
Do not assume a bounded real empirical density.
Track every eta-dependent constant and limiting step.
Return the first uncontrolled integral if closure fails.

`prompts/W4_REAL_FLOW.md`

01 **Input**

Only reviewed upstream lemmas, not their hoped-for extensions.

02 **Work**

Recover singular-force control on actual interacting paths.

03 **Acceptance**

A real bridge; event inclusion alone is insufficient.

W5: recover W_1, then attempt conditional assembly

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

```
Audit the truncated error functional before using W1.
Derive a quantitative particle-tail estimate.
Check the uncut particle flow or regulator limit.
Separate sup_t average(e_i) from average(sup_t e_i).
Use only reviewed inputs in the stability inequality.
If a premise is open, keep the result CONDITIONAL.
Compute the resulting rate instead of assuming A and c.
Do not label a weaker metric as the original target.
```

`prompts/W5_METRIC_ASSEMBLY.md`

01 **Input**

The exact functional and available moment estimates.

02 **Work**

Pay for tails, existence and every limit before Gronwall.

03 **Acceptance**

An honest closure theorem with all premises identified.

The tail estimate cannot be hidden in the last line

A small bounded cost need not imply a comparably small unbounded transport cost.

$$\frac{1}{N}\sum_i e_i \le L\frac{1}{N}\sum_i \min(e_i,1)+\frac{1}{N}\sum_i e_i \mathbf{1}_{\{e_i>L\}}, \qquad L\ge 1$$

ONE POSSIBLE INPUT

Uniform support

A genuine particle support bound can turn bounded cost into W_1, but it must be proved.

A DIFFERENT INPUT

Moment control

A finite moment may control tails only with a rate loss. Optimize L and display that loss.

The particle tail, not just the reference support, is relevant here.

Independent verifier: audit the exact implication

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Read the frozen statement and the complete proof.
Recompute normalization and the Hoeffding projection.
Check annuli, zero speed and impact-parameter limits.
Audit static versus adaptive selection.
Inspect microscopic scales, tails and cutoff removal.
Reject circular use of the final closeness estimate.
Report the first gap; do not silently fill it.
Preserve the native verdict and the exact review scope.

`prompts/VERIFIER.md`

01 **Fresh context**

Reduce the influence of the author's success narrative.

02 **Specific objections**

Name the equation, hypothesis and failed inference.

03 **Separate review levels**

An LLM pass is still not a formal proof certificate.

Launch Danus through real project commands

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
# From the installed Danus repository
./bin/codex
# Paste prompts/DANUS_MAIN.md into Main.

./bin/danus list --json
./bin/danus status vlasov-a1 --json
# Use actual worker IDs returned by the runtime.
# Do not start every worker just to fill the roster.
```

Review paths, permissions and model settings before execution.

01

Initialize

Let Main read the installed contracts and operator settings.

02

Assign

The CLI replaces TASK.md; archive the previous task first.

03

Monitor

Check per-worker status and the raw review artifacts.

A useful checkpoint names the remaining obstacle

Report mathematical changes, not just activity counters.

```
PROJECT: vlasov-a1
NEW_REVIEWED_RESULTS: ...
REJECTED_CLAIMS: ...
FIRST_UNRESOLVED: equation ...
DEPENDENCY_CHANGES: ...
NEXT_MINIMAL_TASK: ...
LIVE_WORKERS: ...
ACTUAL_ROUTE_AND_USAGE: ...
```

A real checkpoint

“The mean estimate is reviewed. The moving-subset bound is still open. Next: test whether the claimed time-selection inequality fails for two switching rows.”

Rethlas prompt: keep the first run local

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

```
Task: CT-FREE-01. Read the sibling .refs/ contract.
Work only with iid initial data and free transport.
Define the pair occupation kernel exactly.
First derive geometry; then mean; then variance.
Keep velocity balls and annuli distinct.
Do not assume subset uniformity or the final N-rate.
Submit one complete local statement for review.
Export a handoff tagged FREE-FLOW ONLY.
```

`prompts/RETHLAS_CT_FREE_01.md`

01

Freeze

The global theorem is motivation, not a premise.

02

Sequence

Each calculation earns the input to the next one.

03

Export

The result can be independently rechecked inside Danus.

R0: derive the geometry before probability

A line crossing a ball is explicit; the nonlinear-flow extension is not.

$$|I \cap \{t : |r_0 + tw| \le R\}| \le \min\left\{|I|, \frac{2R}{|w|}\right\}, \quad w \neq 0$$

Input	r_0, w, R and I = [a,b]. Keep the impact parameter and closest-approach time.
Calculation	Solve $\lvert r_0 + tw\rvert^2 \le R^2$; intersect the resulting interval with I.
Boundary cases	w = 0, tangency, empty intersection and an endpoint at closest approach.
Deliverable	An exact formula plus the fast/slow regime table; test code only supports the formula.

The bound 2R/U needs |w| > U. It is false as a uniform bound on the ball |w| ≤ U.

R1: prove a mean bound for the pair occupation

Start with deterministic R, U and I; do not begin with arbitrary adaptive weights.

$$h_{R,U,I}(z,z') = \int_I \mathbf{1}_{\{|x-x'+t(v-v')|\le R\}}\,\mathbf{1}_{\{|v-v'|\le U\}}\,dt$$

$$\Gamma_N = \frac{1}{N^2}\sum_{i\neq j} h_{R,U,I}(Z_i^0, Z_j^0)$$

$$\mathbb{E}\,\Gamma_N \le C(f_0,T)\,|I|\,R^3U^3$$

Use the bounded free-transport density and the six-dimensional box volume R^3U^3.
Keep the factor $(N-1)/N$ and the dependence on f_0 and T.

R2: derive the variance, not an independence shortcut

Pairs (i,j) and (i,k) share an input and are generally dependent.

$$\Gamma_N = \frac{N-1}{N} U_N, \qquad U_N = \binom{N}{2}^{-1} \sum_{i<j} h(Z_i, Z_j)$$

$$\mathrm{Var}(U_N) = \frac{4}{N} \mathrm{Var}(h_1(Z)) + \frac{2}{N(N-1)} \mathbb{E}\left[h_2(Z, Z')^2\right]$$

ORDER 1/N

One-point projection

$h_1(z) = E[h(z,Z')] - \theta$.
This term is generally nonzero.

ORDER $1/N^2$

Degenerate remainder

$h_2 = h - \theta - h_1(z) - h_1(z')$.
Estimate its own second moment.

R3: sum only what has actually been proved

A good fixed-scale remainder is not automatically usable at every microscopic scale.

$$\sum_{R\in\mathcal{R}_N}\sum_{U\in\mathcal{U}_N}\left(R^{-1}\mathrm{Rem}_N^{\mathrm{near}}+R^{-2}\mathrm{Rem}_N^{\mathrm{far}}\right)$$

Write the scale set	Specify the lowest R, the velocity partition and the time intervals.
Carry the real weights	Include error-class weights and any residence-time gains before simplifying.
Price uniformity	Record the number of events and the exact failure probability after union bounds.
Return the boundary	Report convergence, logarithmic loss or divergence; do not presuppose the target rate.

If this sum fails, return the precise obstruction. Do not hide it behind “standard dyadic summation”.

Rethlas repair instruction: fix the first failure

Copy the task contract; the right-hand column explains its purpose.

COPYABLE PROMPT

Read the raw rejection and freeze the failed statement.
Reproduce the earliest obstruction independently.
Try one minimal repair under the same hypotheses.
Show the new inequality and its parameter cost.
A new assumption creates a separate weaker lemma.
Resubmit only a complete repaired local proof.
After two unchanged failures, return the exact gap.
Do not replace progress with a renamed proof outline.

`prompts/RETHLAS_REPAIR.md`

01 **Diagnose**

Work before the first invalid implication, not after it.

02 **Measure change**

A repair must improve an estimate or alter a justified mechanism.

03 **Stop locally**

A finite local budget is not a claim that the problem is unsolvable.

Handoff: one frozen packet, two review contexts

The frameworks exchange evidence files, not credentials or informal success labels.

Rethlas	Freeze	Danus
Local candidate and raw review	Statement, proof, constants, hash	New submission under project scope

Must travel	Flow type, parameter ranges, predecessor statements, source versions and remaining gaps.
Must not travel	API keys, hidden credential files, or a false assumption that “verified” is a formal certificate.
Acceptance	Danus reviews the exact imported claim. No direct copying into its fact graph.

Use templates/HANDOFF.json. A hash identifies the candidate that was checked, not its correctness.

A milestone ladder keeps partial results honest

Each rung is useful; none should be renamed as the rung above it.

M1 Local identity	Exact free-flow geometry and degenerate cases.
M2 Fixed-scale law	Mean/variance or concentration with all parameters visible.
M3 Uniformity	The required selection class and weighted scale sum.
M4 Dynamics	A genuine singular-force bridge, particle tails and regulator control.
M5 Main target	Only after all inputs pass: uniform-in-time W_1, derived rate and probability.

A weaker N-exponent or bounded metric must remain explicitly weaker than the original target.

Set the deployment boundary before installation

Repository scripts may run autonomous shell commands with your user privileges.

01

Host isolation

Use a dedicated, disposable Linux VM or container for Danus. A separate folder is not a sandbox.

02

Local services

Bind verification to loopback and avoid port collisions. Check both service and MCP endpoints.

03

Secrets and cost

Keep credentials out of Git, prompts and logs. Authorize real calls and enforce limits outside prompts.

The upstream Rethlas runner and verifier include sandbox-bypass flags. Review them before running.

Install Rethlas ①: verification service

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
git clone https://github.com/frenzymath/Rethlas.git
cd Rethlas
git rev-parse HEAD
cd agents/verification
python3 -m venv .venv
source .venv/bin/activate
python -m pip install -r requirements.txt
python -m uvicorn api.server:app \
  --host 127.0.0.1 --port 8091
```

Review paths, permissions and model settings before execution.

01 **Before starting**

Install/authenticate Codex and review the service launcher.

02 **Verifier settings**

Set CODEX_MODEL and CODEX_REASONING_EFFORT to supported values.

03 **Remember**

HTTP health does not prove model authentication works.

Install Rethlas ②: run your local Vlasov task

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
# From Rethlas/agents/generation
python3 -m venv .venv
source .venv/bin/activate
python -m pip install -r mcp/requirements.txt
# Copy the task to data/vlasov_ct_free_01.md
# Put PROBLEM.md in data/vlasov_ct_free_01.refs/
PROBLEM_FILE=data/vlasov_ct_free_01.md \
  MAX_ITERATIONS=2 bash tests/run_example.sh
```

Review paths, permissions and model settings before execution.

01 **Separate variables**

Generation uses MODEL / REASONING_EFFORT, unlike the verifier.

02 **Relative path**

PROBLEM_FILE must be under data/ and relative to generation/.

03 **Read outputs**

Keep blueprint, logs, memory and the actual verification report.

Installation succeeds only after a real smoke test

Run a tiny known proof and a tiny known error before spending on PDE research.

HTTP

The service is listening

›

Model call

Authentication and routing work

›

Review behavior

A plainly invalid proof is rejected

Health probe	curl -f http://127.0.0.1:8091/health
Toy pair	Check n(n+1) is even; separately submit the false claim that every integer is even.
Evidence	Save raw responses and logs. A smoke test is not a reliability theorem for the verifier.

Install Danus ①: use the codex branch

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
git clone --branch codex \
  https://github.com/frenzymath/Danus.git
cd Danus
git rev-parse HEAD
# Read docs/security-and-trust.md first.
bash scripts/bootstrap.sh
cp -n config/danus.env.example config/danus.env
cp -n config/codex.env.example config/codex.env
```

Review paths, permissions and model settings before execution.

01 **Platform**

The documented bootstrap expects Linux and provisions runtime/.

02 **Branch**

The codex branch runs Main on Codex; do not mix main-branch instructions.

03 **Preserve config**

Do not overwrite existing credentials or a working runtime.

Install Danus ②: choose one authentication route

A ChatGPT login and a provider API key are different credential and billing paths.

CHATGPT LOGIN

```
# config/danus.env
CODEX_BACKEND=chatgpt

bash scripts/setup-codex.sh login
bash scripts/check-codex.sh
```

RESPONSES-COMPATIBLE API

```
# config/codex.env
CODEX_BACKEND=api
# Fill endpoint, model and key safely.

bash scripts/setup-codex.sh api
bash scripts/check-codex.sh
```

check-codex.sh may make a live API call on the API route. Review authorization and cost first.

Install Danus ③: service, diagnostics and Main

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
# Example: use 8093 if Rethlas already uses 8091.
# Set VERIFY_PORT=8093 in config/danus.env.
bash scripts/services.sh up verify
bash scripts/check-codex.sh
bash scripts/doctor.sh
./bin/codex

# Initialize; then paste the Vlasov Main contract.
```

Review paths, permissions and model settings before execution.

01 **Service**

No proof fact is admitted without the verifier gate.

02 **Pilot limits**

Start with two rounds and a small per-round timeout, then inspect results.

03 **Next action**

Run a toy submission before launching CT-FREE-01.

macOS: diagnose platform mismatch, do not hide it

A Linux binary cannot be repaired by changing its executable permission.

```
uname -sm
command -v node npm python3 codex
node --version
python3 --version
file runtime/node22/bin/node
command -v setsid
./bin/codex --version
```

Binary format

Check Mach-O versus Linux ELF and the CPU architecture.

Runtime mismatch

System Codex may work while the repository wrapper fails.

Practical route

Prefer a dedicated Linux environment; audit a native macOS port separately.

Pharos installation and first session

Manual setup on a dedicated Linux host, using ChatGPT subscription login.

TERMINAL

```
git clone https://github.com/frenzymath/Pharos.git
cd Pharos
bash scripts/bootstrap.sh
cp -n config/pharos.env.example config/pharos.env
# Set CODEX_BACKEND=chatgpt in config/pharos.env.
bash scripts/setup-codex.sh login
source scripts/env.sh
pharos doctor
codex
```

For an API or Azure backend, use the configuration in the official setup guide.

01

Prepare the host

Use Linux with Python 3.10+, Git, Bash, curl, tar and tmux.

02

Check the setup

Run pharos doctor. Use Chrome or Chromium for reports and TeX for papers.

03

Open root Ops

Run codex in the repo root. Give Ops the question and the paths to your materials.

DeepSeek API: match endpoint, model and credentials

Model names and capabilities must come from the current provider documentation.

BASE URL

Endpoint

https://api.deepseek.com

Use the API base URL, not the web chat address.

MODEL ID

Model

deepseek-flash
deepseek-v4-pro

Verify the account’s available models.

SECRET

Credential

DEEPSEEK_API_KEY

Read from environment or hidden input; never paste a live key into slides.

An OpenAI-compatible SDK request to DeepSeek uses DeepSeek credentials and billing.

Begin with a tiny API call, not a full paper

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
python3 -m venv .venv
source .venv/bin/activate
python -m pip install -U openai
export DEEPSEEK_MODEL=deepseek-flash

python scripts/deepseek_smoke.py \
  --mode chat --allow-paid
python scripts/deepseek_smoke.py \
  --mode responses --allow-paid
```

Review paths, permissions and model settings before execution.

01

Least data

Send a one-word test, not unpublished research.

02

Two protocols

Chat success alone does not validate the Responses tool chain.

03

Record safely

Keep model, request ID and usage; never serialize the key.

Python: one client, two protocol checks

These are compact excerpts; use the packaged script for bounded retries and error handling.

CHAT COMPLETIONS

```
from openai import OpenAI
client = OpenAI(
    api_key=key,
    base_url="https://api.deepseek.com")
r = client.chat.completions.create(
    model=model,
    messages=[{"role": "user",
               "content": "Reply OK."}],
    max_tokens=256)
print(r.choices[0].message.content)
```

RESPONSES

```
r = client.responses.create(
    model=model,
    input="Reply OK.",
    max_output_tokens=256)
print(r.output_text)

# key comes from environment/getpass.
# Check tool calls separately.
# Do not assume server-side state.
```

Connect DeepSeek to Codex using the provider guide

Copy the task contract; the right-hand column explains its purpose.

TERMINAL

```
mkdir -p ~/pde-setup
cd ~/pde-setup
SETUP="https://cdn.deepseek.com/api-docs"
curl -fL "$SETUP/codex-deepseek-setup-en.sh" \
  -o codex-deepseek-setup.sh
less codex-deepseek-setup.sh
# Review changes before execution.
bash codex-deepseek-setup.sh
```

Review paths, permissions and model settings before execution.

01 **What changes**

Provider, model catalog, wire_api, authentication and capability metadata.

02 **Protect backups**

The guide's manual setup can store a token in config; treat backups as secrets.

03 **Correct home**

Do not assume ~/.codex is the same home used by Danus.

Danus + DeepSeek: align the entire route

A wiring template, not a claim of live compatibility with every installed version.

config/codex.env

```
CODEX_BACKEND=api
CODEX_API_BASE_URL=\
"https://api.deepseek.com"
CODEX_API_MODEL="deepseek-flash"
DANUS_CODEX_API_KEY="$DEEPSEEK_API_KEY"
```

config/danus.env

```
DANUS_MAIN_MODEL="deepseek-flash"
DANUS_MAIN_EFFORT="high"
DANUS_WORKER_MODEL="deepseek-flash"
DANUS_VERIFY_MODEL="deepseek-flash"
DANUS_VERIFY_EFFORT="high"

# Verify worker effort and model catalog.
# Re-run setup, health, then toy proof.
```

Mixed models require separate, verified routes

Two model names do not automatically configure two providers.

Explore

DeepSeek or another authorized model

›

Freeze

Statement + proof + dependencies

›

Review

GPT on an independently checked route

Verify every role	Actual binary, CODEX_HOME, provider, URL, credentials, model and supported effort.
Start simply	Separate deployments and a manual frozen-file handoff are easier to audit than silent routing.
No silent fallback	Report a provider failure and any authorized fallback explicitly in the run record.

Troubleshoot the layer that actually failed

Use bounded retries, redact secrets and keep TLS verification enabled.

Symptom	Inspect first	Do not do this
401 / auth	Provider, key and inherited environment	Paste credentials into a report
400 / parameters	Model, protocol, effort and catalog	Copy xhigh/max across providers
429 / limits	Concurrency, token limits, retry policy	Increase workers and retry forever
Health OK, proof fails	Real model call, MCP and verifier logs	Treat HTTP 200 as a proof pass
Busy, no new result	First gap and changed mathematical evidence	Rename the same failed task

The first pilot: close one evidence loop

An execution plan, not a promise of a theorem within a fixed wall-clock time.

Prepare

Contract, versions and permission

›

Explore

Geometry, probability, adversary

›

Review

One frozen local claim

›

Decide

Repair, weaken or expand

Rethlas first output	The exact free-flow identity, mean/variance ledger and first summability obstruction.
Danus first output	Reviewed local claims, rejected shortcuts and the next minimal research task.

Do not launch the full endpoint theorem until the local gate has produced genuinely new evidence.

Close one lemma before assembling a paper

Good research automation keeps the missing step visible.

01 Freeze the statement.

02 Make the computation reproducible.

03 Review the exact proof independently.

04 Keep all open dependencies visible.

References ①: OpenAI interfaces and execution

Use the installed CLI help and current official documentation.

O1 **Codex CLI**
https://developers.openai.com/codex/cli

O2 **ChatGPT Work**
https://learn.chatgpt.com/docs/get-started-with-work

O3 **Subagents**
https://learn.chatgpt.com/docs/agent-configuration/subagents

O4 **AGENTS.md**
https://learn.chatgpt.com/docs/agent-configuration/agents-md

O5 **Non-interactive Codex**
https://learn.chatgpt.com/docs/non-interactive-mode

O6 **Authentication**
https://learn.chatgpt.com/docs/auth

References ②: Rethlas and Danus

Original figures are reproduced from the user-supplied files; model labels remain historical.

R1 **Rethlas repository, main branch**
https://github.com/frenzymath/Rethlas

R2 **Rethlas generation runner**
https://github.com/frenzymath/Rethlas/blob/main/agents/generation/tests/run_example.sh

R3 **Rethlas verification service**
https://github.com/frenzymath/Rethlas/blob/main/agents/verification/api/server.py

R4 **Rethlas paper, Figure 2**
https://arxiv.org/html/2604.03789v2

D1 **Danus repository, codex branch**
https://github.com/frenzymath/Danus/tree/codex

D3 **Danus security and trust**
https://github.com/frenzymath/Danus/blob/codex/docs/security-and-trust.md

D8 **Danus CLI and tools**
https://github.com/frenzymath/Danus/blob/codex/docs/cli-and-tools.md

D9 **Danus paper, Figure 1**
https://arxiv.org/html/2607.06447v2

References ③: deployment, APIs and Vlasov

Pin repository commits and article versions; no novelty claim or main theorem is certified here.

D2 **Danus getting started**
https://github.com/frenzymath/Danus/blob/codex/docs/getting-started.md

D4 **Danus backend configuration**
https://github.com/frenzymath/Danus/blob/codex/config/codex.env.example

D5 **Danus runtime configuration**
https://github.com/frenzymath/Danus/blob/codex/config/danus.env.example

D6 **Danus operating guide**
https://github.com/frenzymath/Danus/blob/codex/docs/operating-guide.md

D7 **Danus bootstrap source**
https://github.com/frenzymath/Danus/blob/codex/scripts/bootstrap.sh

A3 **DeepSeek official Codex integration**
https://api-docs.deepseek.com/quick_start/agent_integrations/codex/

M1 **Hauray and Jabin, singular-force propagation of chaos, v6**
https://arxiv.org/abs/1107.3821v6

M2 **Fournier and Guillin, empirical Wasserstein convergence**
https://arxiv.org/abs/1312.2128

References ④ Pharos and ChatGPT Dots

Official architecture, setup and operating guides.

P1 **Pharos architecture and original diagram**
https://github.com/frenzymath/Pharos/blob/main/docs/architecture.md

P2 **Pharos installation and first session**
https://github.com/frenzymath/Pharos/blob/main/docs/getting-started.md

P3 **Pharos operating guide**
https://github.com/frenzymath/Pharos/blob/main/docs/operating-guide.md

O7 **ChatGPT Dots overview and availability**
https://learn.chatgpt.com/docs/dots

O8 **Dots setup**
https://learn.chatgpt.com/docs/dots/getting-started

O9 **Dots task delegation and Activity**
https://learn.chatgpt.com/docs/dots/tasks-and-memory