Inside the Verified Proof: A Step-by-Step Walkthrough of the Lean Navier–Stokes Formalization

Bevan Hunt

DOI: PENDING CC BY 4.0

Self-published, no affiliation. Released under the Creative Commons Attribution 4.0 International License (creativecommons.org/licenses/by/4.0).

repository: https://github.com/truemagic-coder/ns-proof

Abstract

This paper walks through the Lean code of the Navier–Stokes/Euler formalization as it was verified: the repository structure, the trust model of a prebuilt-`.lake` replay, the theorem surface, the torus machinery behind the (8.19) estimate, the L^3 comparison machinery of §10.3, the Euler side, and the two repair commits (the (8.19) formalization and the corrected $(10.19)'/(10.17)'$ estimates). Every code listing is verbatim from the repository at a pinned commit. The companion mapping paper [1] tells you where each piece is and what the machines checked; this paper tells you what the code does.

Part I

The repository, and what "verified" means here

The object of verification is a Lean 4.34.1 repository: `NavierStokesAndEuler-fork` – the OpenAI base repository `NavierStokesAndEuler` at `f9e8bc5` (Lean v4.34.0-rc2) updated to 4.34.1 with prebuilt dependencies, plus the pilot's repair commits.

Technical lane — the real math

Trust model. The verification job does not recompile the dependencies. The job workspace carries a fully prebuilt `.lake` (olean files for mathlib, Comparator, Lean4Export, and the workspace itself, with the `.git` checkouts of the `.lake/packages/<dep>` dependencies that `lake` needs to validate the checkouts against the lakefile). `lake build` in the job then *replays*: the Lean kernel re-verifies every olean against its source hash and proof term in the checking environment; only the challenge files are built fresh. A green build means every proof term was re-checked by the kernel – the strongest verification available short of running the kernel twice.

Checkpoint 1 `cp-model`

Evidence: the signed verification report (job `31974058-a665-472d-b6f8-00e4ec9e2d45`): criterion ***`build`*** *PASS – the replay build re-verified all oleans; the SBOM records the dependency manifest*

Technical lane — the real math

Axiom policy. The permitted axioms are `propext`, `Classical.choice`, and `Quot.sound` – the standard library's own. The `result_audit_ns` criterion greps the built olean surface for axiom declarations beyond that set; the signed report records `axioms` (empty beyond the permitted set) as part of the verdict.

The directory layout of the theorem code:
Lean — repository layout (theorems only)

Part II
The theorem surface

The target theorem, verbatim – this is what the reference checkers re-check:
Lean — NavierStokes/ComparatorSolution.lean:16

The proof is a single line: the theorem's *content* is discharged by `ComparatorBridge`, the bridge module that connects the statement to the constructed blow-up solution and the §10.3 comparison argument. The bridge is where the paper's §10.3 lives in Lean.
Lean — NavierStokes/R3/WholeSpaceUniqueness.lean:30

The difference-energy bound that the closure of §10.3 needs: the weighted L^2 norm of $v - u$ with the cutoff χ_R stays bounded. The full statement (fork file/line in [1], mapping table row `10.3-diffenergy`) carries the hypotheses that both fields solve the equations with the same force and initial data, and the uniform regularity bounds on u.

Technical lane — the real math

The closure chain, as the code organizes it:

1. `exists_weighted_difference_energy_identity` – the energy identity with the recovered pressure (the (10.16) side);
2. `exists_commutator_pressure_flux_bound` – the commutator bound (the (10.18b) side, $R^{-3/4}$-type);
3. `exists_uniform_actual_pressure_flux_bound` – the *actual* flux bound with the A_R-dependent right-hand side (the code's version of the (10.18a) pairing – *not* the page's (10.19));
4. the absorption: Young's inequality on the A_R powers (all $\leq 3/2 < 2$ under the code's bound), dissipation absorbs, Gronwall;
5. the $R \to \infty$ limit and the difference-energy identity give $w \equiv 0$; uniqueness; contradiction with the constructed blow-up.

Checkpoint 2 `cp-surface`

Evidence: the target theorems at `NavierStokes/ComparatorSolution.lean:16` and `Euler/Solution.lean:33` – comparator re-checked in the signed report (criteria `comparator_ns`, `comparator_euler`, PASS)

Part III

The torus machinery (the (8.19) code)

The (8.19) estimate – "r derivatives of the inverse are bounded by $r+4$ input derivatives" – lives in the smooth-family torus-inverse machinery. The code's main route is the $m+5$ version; the page's $m+4$ version is a corollary that the pilot formalized (Part VI).

Lean — NavierStokes/SmoothFamilyTorusInverse.lean:1081 (the code's own m+5 route)

The input hypothesis is rapid decay at rate $m+1$; the output bounds the Fourier-derivative word of length r of the inverse g. The proof's engine is the coefficient machinery of `SmoothFourierData`:

Lean — NavierStokes/SmoothFourierData.lean:361 (the rapid-decay summability)

Technical lane — the real math

The key analytic lemma in the code: for a product majorant $(1+|k|)^{-(m+1)}$ with $m \geq r+4$-type weights, the convolution-type sum over the lattice splits symmetrically at $|n| \approx |k|$ into two tails, each summable by the $(1+|n|)^{-3/2}$ majorant (the $d = 1$ lattice is summable at exponent > 1; $3/2$ is the code's uniform exponent, and the $(1+|k|)$ weight carries the rest). This is exactly the page's (8.19) mechanics – the symmetric split – and it is what makes the $m+4$ route available: the $r+4$ derivative budget pairs with the -3 decay in the page, and the code's uniform $-3/2$ majorant covers it.

Checkpoint 3 `cp-torus`

Evidence: the (8.19) route in the fork: `NavierStokes/SmoothFourierData.lean:361,381,413` (the majorant, the symmetric split, the lattice sum); the full walk-through in [1], §2.2

Part IV

The R3 comparison machinery

The §10.3 comparison argument, as the code organizes it, lives in `NavierStokes/R3/`. The cutoff geometry is the shared substrate:

Technical lane — the real math

`ComparisonCutoffs.lean` fixes the smooth cutoff ϕ (1 on the unit ball, supported in B_c) and the rescalings $\phi_\lambda(x) = \phi(x/\lambda)$, with the support inclusions and the gradient estimate $|\nabla\phi_\lambda| \leq C\lambda^{-1}\phi_c'\lambda$ that every flux bound uses. The annuli: $\operatorname{supp}\nabla\chi_R \subset [8R, 8cR]$ with $\chi_R = \phi_8R$. (The page's support bookkeeping errors, [**?**] §2.3, are errors *against* this geometry.)

The flux estimate the code proves – the object of the whole alignment question:

Lean — NavierStokes/R3/PressureFlux.lean:584 (the A_R-dependent bound – the code's argument)

Technical lane — the real math

Reading the bound: the flux through $[8R, 8cR]$ is controlled by $(B_R^{1/2} + 1)(A_R/R + R^{-2}) + R^{-7/4}B_R^{3/4}$ – the L^6-average on the *inner* ball, multiplied by the *dissipation* A_R (the energy of ∇w) and the gradient of the cutoff, R^{-1}. Compare with the page's (10.19), $C_TR^{-1}(B_R + 1)B_R^{1/2} + R^{-3/4}B_R^{3/4}$: the page has no A_R, and its derivation's pairing is invalid ([**?**]). The code's bound is what the closure consumes: under (10.17) every power of A_R is $\leq 3/2 < 2$, so Young's inequality absorbs the flux into the dissipation with room to spare.

Checkpoint 4 `cp-flux`

Evidence: `NavierStokes/R3/PressureFlux.lean:584` *– kernel-verified and comparator-checked; the divergence from the page's (10.19) documented in [1] mapping table row* `10.3-flux`

Part V
The Euler side

The Euler companion – the same construction for the inviscid equations – shares the torus machinery and has its own target:

Lean — Euler/Solution.lean:33

Technical lane — the real math

The Euler singularity statement mirrors the NS one: a constructed divergence-free smooth solution with bounded energy that cannot be extended smoothly – and the non-existence of any smooth bounded-energy extension. The pilot's alignment check (report leg 3) found the Euler *statements* faithful to the paper's claims within the checked scope: no mistranslation at lemma level. The Euler comparator target (`Euler/Solution.lean:33`) was re-checked in the signed report (criterion `comparator_euler`, PASS).

Checkpoint 5 `cp-euler`

Evidence: `Euler/Solution.lean:33` *and* `Euler/EulerSingularity.lean:--`*; the alignment leg: report leg 3, "no mistranslation found within checked scope"*

Part VI
What we added, and what the kernel decided

Two commits, both kernel-verified. The first closes the (8.19) gap; the second formalizes the repair of the broken argument.

Technical lane — the real math

Commit `9bbfa7b` *– (8.19) as written.* The new lemma `NavierStokes/SmoothFamilyTorusInverse.lean:1109`:

`norm_derivativeWord_inverse_le_m4` : $m \geq r{+}4 \;\Rightarrow\; \|\partial^{(r)} g\|_\infty \leq C_r(1{+}r)^{O(1)} \sup_k(1{+}|k|)^{r+4}|\hat{f}(k)|.$

The proof reuses the $m{+}5$ route's decomposition and re-tightens the weight by one derivative: the $r{+}4$ budget pairs with the summable $(1{+}|k|)^{-3}$ majorant (the symmetric split of `SmoothFourierData`), so the extra derivative the code's route carries is not needed for the page's claim. The kernel accepted it on the first attempt. Consequence: *the page's (8.19) is sound* – the Lean kernel checked it.

Checkpoint 6 `cp-819`

Evidence: fork commit `9bbfa7b` *–* KERNEL-ACCEPTED*; the formalization walk-through in [1], §2*

Technical lane — the real math

Commit `24b1944` *– the corrected estimates.* The new file `NavierStokes/R3/Paper1019Repaired.lean:187--240`:

- `exists_repaired_pressure_flux_bound` ($(10.19)'$): the flux bound with the L^6 norm measured on a ball that *covers the annulus* (ϕ_8cR instead of ϕ_4R) and the A_R-dependent right-hand side the geometry requires;
- `repaired_sobolev_covering` ($(10.17)'$): the matching Sobolev inequality for the covering ball, so the closure's substitution still works;
- `repaired_closure_sketch`: the absorption step under the corrected bound – Young's inequality, powers $\leq 3/2 < 2$, dissipation absorbs.

The kernel accepted all three. What is *not* in the commit: the full Gronwall/$R \to \infty$ chain with $(10.17)'$ substituted – the open row of the mapping table. Until that is formalized, the repaired argument's closure is a reading, not a verdict.

Checkpoint 7 `cp-repair`

Evidence: fork commit `24b1944` *–* `NavierStokes/R3/Paper1019Repaired.lean:187--240`*,* KERNEL-ACCEPTED*; the open closure row in [1], §4–5*

Technical lane — the real math

The verification job. The independent gate: job `31974058-a665-472d-b6f8-00e4ec9e2d45` against the fork at `24b1944` – the prebuilt-`.lake` replay build (Part I's trust model), the two comparator targets re-checked by the reference kernels (landrun + nanoda), the axiom audit, and the SBOM. Verdict: `accepted` (signed report, [1], §3). That verdict says: the Lean object proves itself under `propext` + choice, re-checked by a second kernel. It does *not* say: the paper's argument is correct. The gap between those two sentences is the subject of this series.

Checkpoint 8 `cp-job`

Evidence: the signed verification report, job `31974058-a665-472d-b6f8-00e4ec9e2d45`*:* `accepted` *– build* PASS*, comparator_ns* PASS*, comparator_euler* PASS*, audit* PASS*; the report's criteria table and hash fields*

Where to go from here. The open row: the full closure with $(10.17)'$. The code for the Gronwall step exists (the original closure, built for the code's A_R-bound); the work is the substitution and the $R \to \infty$ limit under the covering-ball geometry. If it formalizes, the repaired argument closes and the one-liner sharpens: the paper's argument can be repaired, and the repair is verified. If it obstructs, the obstruction is the next candidate for the counterexample program of [**?**].

References

[1] Bevan Hunt. The proof as the machine checked it: The verified Lean formalization of the navier–stokes blowup argument. Companion paper, this series, 2026. DOI: PENDING.