

TECHNICAL REPORT · VERIFIED DIFFERENTIABLE CFD IN RUST

Honest Fluids: A Verified, Differentiable Computational Fluid Dynamics Stack for Physical AI, in the Browser

One open Rust stack spanning every solver paradigm — each checked against an analytic or reference oracle, differentiable end to end, and deployed self-verifying to WebAssembly.

David Jean Charlot, PhD

The Charlot Lab, Institute for Physical AI @ BMI

Correspondence: contact@physicalai-bmi.org · physicalai-bmi.org · Interactive companion: physicalai-bmi.org/fluids

CC0 1.0 Universal — dedicated to the public domain. Reference implementation: [ferromotion-fluid](#) (Rust).

*Deployed robots do not run computational fluid dynamics in the control loop, and the discipline's own 2026 consensus is that verification — not throughput or model size — is its weak layer. We report a fluid-dynamics stack that treats verification as the primitive: every solver ships with an oracle it must reproduce before it is admitted. The stack spans all five solver paradigms — pressure-projection (MAC), lattice-Boltzmann (D2Q9/D3Q19, CPU and GPU), smoothed-particle hydrodynamics with volume-of-fluid, Fourier-spectral Navier–Stokes, compressible finite-volume (HLL/Sod), and structured and unstructured finite volume up to a coupled Taylor–Hood incompressible Navier–Stokes solver. Each is verified against an analytic or reference solution (Ghia lid-driven cavity, Taylor–Green decay, the exact Riemann problem, the Method of Manufactured Solutions) at the tolerances tabulated in §3. The stack is differentiable by construction: gradients through the lattice, through the global pressure projection (a self-adjoint solve reused backward), through the fluid–structure loop, and through the unstructured sparse assembly all match finite differences to machine precision (§4). Two products follow from verification-first design: a **surrogate honesty harness** that reads the physics directly and flags a prediction which shares a competitor's mean-squared error but violates incompressibility or the energy spectrum (§5); and a **coefficient-model diagnostic** that shows exactly where lumped drag breaks under a gust and reconstructs the classical unsteady-force hierarchy from the resolved solver (§6). The entire stack is pure Rust, hardware-unbound (CPU floor, GPU across silicon through a portable compute layer, WebAssembly in the browser), permissively licensed, and — as far as we can establish — the only differentiable CFD in Rust and the only browser-deployable differentiable fluid stack in existence. It runs, and self-verifies, in a web page.*

1. Position: nothing is owned

The premise is that no capability in fluid dynamics is enclosable, because every incumbent is bound to something — a hardware platform (OpenCL, CUDA), a license (non-commercial GPU codes), a runtime (Python/HPC), or a maintenance cliff (single-maintainer cultures; abandoned differentiable

frameworks). The open architecture is the lever: pure Rust, portable across every silicon, wasm-clean to the browser, permissively licensed, verification-first, differentiable. Three facts fix the strategy honestly.

Fact 1 — deployed robots do not run CFD in the loop, and that is the design point. Real CFD lives in design/certification and in environment precomputation. Control loops everywhere run lumped coefficients plus *learned residuals*; adaptation of the Neural-Fly class beats CFD-in-the-loop for gust rejection [8]. A stack premised on "robots will run CFD onboard" fights the evidence. Ours builds the seam those systems actually live on (§6).

Fact 2 — the capital is in design surrogates, not robotics fluids; and foundation-models-for-physics are honest research with open limits — the training data for a general one has not been paid for. Marketing that pairs "physical AI" with in-loop CFD surrogates is largely aspirational.

Fact 3 — the trust layer is the acknowledged weak point. Vendor surrogate tooling now ships "don't trust the model, test the physics" checks; vanilla physics-informed networks are consensus-unfit for forward turbulence; neural-operator spectral bias is a known, actively-patched failure mode. Verification is where the field admits it is thin — and therefore where an open stack should plant its flag.

From these follows the empty niche we occupy. Every 2026 browser fluid demo is eye-candy without physics accountability; every verified solver is install-only. The intersection — a solver that renders *and self-verifies in the page* — was empty. It is now the interactive companion to this report.

2. Verification as the primitive

The governing equations are the incompressible Navier–Stokes system (and, for the compressible regime, the Euler conservation laws):

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \quad \nabla \cdot \mathbf{u} = 0.$$

The discipline of the stack is that no discretization of this system is admitted until it reproduces a known answer. We use four classes of oracle, in order of strength: (i) **analytic solutions** — Taylor–Green decay $\mathbf{u} e^{-2\nu k^2 t}$, the shear-wave momentum-diffusion mode, the plane-wave advection–diffusion field, the exact Riemann solution of the shock tube; (ii) **reference tabulations** — the Ghia *et al.* (1982) lid-driven-cavity data [1]; (iii) the **Method of Manufactured Solutions** — choose a field, feed the analytic source it implies, and measure the error's convergence order; and (iv) **conservation and structural invariants** — mass, momentum, and energy to machine precision where the boundary fluxes vanish. Every solver in §3 carries at least one such oracle in its test suite; the numbers reported are those tests' outputs, not claims.

3. The solver spectrum

The stack covers all five CFD paradigms. Each row is a solver, its governing oracle, and the verified result.

SOLVER	PARADIGM	ORACLE	VERIFIED RESULT
MAC projection	elliptic/parabolic	Taylor–Green; Ghia cavity [1]	order 1.99; worst dev. 0.0042

LBM D2Q9 (CPU)	kinetic	Taylor–Green (diffusive scaling)	order 3.7; mass drift <1e-12
LBM D2Q9 (GPU)	kinetic	CPU trajectory (ordering-exact)	1.9e-6; ~3.4 GLUP/s
LBM D3Q19	kinetic (3-D)	shear-wave diffusion; 2-D reduction	order 1.99; 2-D→3-D 4e-16
SPH (WCSPH)	particle	momentum; hydrostatic profile	drift 3e-18; dp/dz to 2.5e-4
VOF (minmod TVD)	interface	solid-body rotation; boundedness	vol. drift 1.4e-17; $C \in [0, 1]$
Spectral operators	spectral	band-limited derivative; Poisson	5e-15 (exponential conv.)
Spectral Navier–Stokes	spectral	Taylor–Green decay (nonlinear = 0)	4.66e-15 (machine)
Euler (HLL)	hyperbolic/compressible	Sod exact Riemann [3]	star p^*, u^*, ρ^* <1%
FVM (structured)	finite volume	plane-wave adv.–diff.; conservation	order 2.00; drift 0.0
FVM (unstructured)	finite volume (triangles)	MMS on a jittered mesh	order 2.08
Coupled NS (Taylor–Hood)	incompressible, coupled	MMS (Stokes + Picard NS)	order 3.86; Picard 6 iters

The lattice-Boltzmann update is the BGK relaxation of the discrete populations f_k toward their equilibrium, followed by streaming:

$$f_k(\mathbf{x} + \mathbf{c}_k, t+1) = f_k(\mathbf{x}, t) + \frac{1}{\tau}(f_k^{\text{eq}} - f_k), \quad f_k^{\text{eq}} = w_k \rho \left(1 + 3 \mathbf{c}_k \cdot \mathbf{u} + \frac{9}{2} (\mathbf{c}_k \cdot \mathbf{u})^2 - \frac{3}{2} \mathbf{u}^2 \right)$$

with $\nu = (\tau - \frac{1}{2})/3$. The GPU path reproduces the CPU reference's collide-then-stream ordering exactly, so the verified CPU solver — not a fresh statistical comparison — is its oracle. On the coupled incompressible problem we avoid the collocated velocity–pressure instability entirely by using the inf-sup-stable Taylor–Hood mixed element (quadratic velocity, linear pressure), which is stable by construction rather than by a Rhie–Chow patch; the steady Stokes saddle system is $\begin{bmatrix} \nu K & -B^\top \\ B & 0 \end{bmatrix}$, and the full Navier–Stokes solution is recovered by a Picard iteration over the Oseen convection operator, converging in six iterations at the Reynolds number tested.

4. Differentiability by construction

A verified solver becomes an instrument for inverse problems only if one can differentiate through it. The stack is differentiable throughout, and every gradient is checked against a central finite difference at the tolerance shown.

Kinetic — generic-scalar dual. The lattice update is a rational polynomial in the populations, so carrying a dual number through the full simulation yields exact gradients. Identifying the viscosity from an energy-decay trace recovers τ to relative error 6×10^{-11} .

Projection — self-adjoint reuse. A MAC step is an explicit predictor followed by a *global* pressure-Poisson projection. The projection $(\mathbf{u}^*) \mapsto (\mathbf{u})$ is linear, parameter-free, and self-adjoint in the staggered inner product, so its vector-Jacobian product is the *same* pre-factored Poisson solve run backward — one solve, no re-derivation. Only the nonlinear predictor needs a hand-transposed stencil. The result reads:

$$\overline{\mathbf{u}^*} = \left(\mathbf{I} + \frac{\Delta t}{h} G L^{-1} D \right)^\top \overline{\mathbf{u}}, \quad L = L^\top \Rightarrow L^{-\top} = L^{-1}.$$

Finite-difference checks: $\partial J / \partial \mathbf{u}_0$ to 1.6×10^{-8} , $\partial J / \partial (\text{lid})$ to 6×10^{-12} , $\partial J / \partial \nu$ to 1.4×10^{-8} ; viscosity identification from an observed field converges to floating-point zero.

Fluid–structure — the coupled sensitivity. A self-propelled undulatory swimmer earns thrust from the fluid with no prescribed forward velocity; the exact $\partial(\text{displacement}) / \partial(\text{gait})$, threaded through the coupled immersed-boundary loop including the body's own free degree of freedom, matches finite differences to 5.9×10^{-11} , and the body learns to swim on that gradient.

Unstructured — the discrete adjoint. For $-\nabla \cdot (k \nabla \phi) = S$ on a triangular mesh, the objective gradient with respect to a per-element conductivity field reuses the same sparse Cholesky factor (self-adjoint), giving the entire gradient field in one extra solve; it matches finite differences to 2.6×10^{-6} . To our knowledge this is the only differentiable CFD implementation in Rust and the only browser-deployable one in existence; the dead differentiable frameworks it succeeds were bound to platforms it is not.

5. The honesty harness

Verification-first design yields a product the field is asking for: an audit that reads the physics rather than the error metric. The claim a mean-squared error cannot see is that two predictions can share an identical error against ground truth while one is physically valid and the other a fluent lie. The harness computes receipts directly — the incompressibility residual $\|\nabla \cdot \mathbf{u}\|$ and a fast (FFT-free) spectral-roughness proxy $\|\nabla^2 \mathbf{u}\| / \|\mathbf{u}\|$ that stands in for the high-frequency energy a neural operator smears — and grades a prediction against the ground truth's own receipts. On an adversarial pair constructed to share an MSE:

CHEAT MODE	SHARED MSE	HONEST RECEIPT	CHEAT RECEIPT	VERDICT
injected divergence	0.106	div 6.9e-15	div 0.941	flagged
high-frequency noise	0.071	rough 84.5	rough 729	flagged

The metric cannot separate the pair; the physics receipt separates it cleanly. A surrogate and its audit ship together. As a concrete surrogate to audit, a Dynamic-Mode-Decomposition operator [6] fit to spectral-NS snapshots — reduced to rank three through the method of snapshots, with the whole computation expressed through the $m \times m$ snapshot Gram matrices — reconstructs the training window to 2.2×10^{-4} and *extrapolates* beyond it to 2.1×10^{-3} ; it learned the dynamics, not a lookup.

6. The deployment seam: where coefficients break

Because deployed control uses coefficients, the honest contribution is to say exactly where they break and how the resolved solver repairs them. Drive a body through a gust with the resolved immersed-boundary solver and fit the classical unsteady-force hierarchy to the measured force:

$$F \approx \underbrace{-\frac{1}{2}\rho C_d A |U|U}_{\text{quasi-steady}} \underbrace{-m_a \dot{U}}_{\text{Morison added mass}} \underbrace{-c_b \int_0^t \frac{\dot{U}(\tau)}{\sqrt{t-\tau}} d\tau}_{\text{Basset history}}.$$

In steady approach the quasi-steady coefficient already fits to a few percent — the coefficient model is not wrong, only incomplete. During the gust it leaves $\approx 94\%$ of the force unexplained; the added-mass term drops the residual to $\approx 7\%$, and the history term halves it again. The resolved solver reconstructs the unsteady-force physics from data — which is precisely how a learned residual [8] is made, measured, and trusted. The same stack closes the loop to the robot: from a divergence-free environment wind field (verified by the harness to 1.6×10^{-14}), a Zermelo minimum-time planner routes a vehicle to ride tailwinds and skirt headwind pockets, beating a wind-naïve straight line by a margin that grows with the wind.

7. Claim

The disclosed object is a design discipline and its reference implementation: *every layer of CFD — solver to surrogate to syllabus — reimplemented open, verified against an analytic or reference oracle, differentiable by construction, and hardware-unbound, in Rust*. Every solver paradigm is present and oracle-checked at the tolerances of §3; the stack is differentiable end to end at the tolerances of §4; the honesty harness discriminates physics violations that error metrics cannot (§5); and the coefficient-break diagnostic supplies the seam deployed systems actually use (§6). The entire stack compiles to WebAssembly and self-verifies in a browser page. No constituent method is enclosable — all are public numerical analysis — and no term of the discipline is bound to a platform, license, or runtime.

Selected prior art

1. U. Ghia, K. N. Ghia, C. T. Shin. High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *J. Comput. Phys.* 48(3):387–411, 1982.
2. A. J. Chorin. Numerical solution of the Navier–Stokes equations. *Math. Comp.* 22:745–762, 1968.
3. E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 3rd ed., 2009 (Sod shock tube; HLL flux of Harten, Lax & van Leer, 1983).
4. J. J. Monaghan. Smoothed particle hydrodynamics. *Rep. Prog. Phys.* 68:1703–1759, 2005.
5. C. S. Peskin. The immersed boundary method. *Acta Numerica* 11:479–517, 2002.
6. P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* 656:5–28, 2010.
7. F. Brezzi, M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer, 1991 (Taylor–Hood; the inf-sup / LBB condition).
8. M. O’Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung. Neural-Fly enables rapid learning for agile flight in strong winds. *Science Robotics* 7(66), 2022.
9. S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.
10. P. J. Roache. Code verification by the method of manufactured solutions. *J. Fluids Eng.* 124(1):4–10, 2002.

AI-use disclosure. Preparation of this technical report used a large language model (Claude, Anthropic) for drafting, editing, typesetting the equations, and building the reference implementation and its interactive companion. The author reviewed the content, verified the cited prior art, and is solely responsible for the work; every numerical result

reported here is the output of an executable verification test in the reference implementation. Consistent with ICMJE, COPE, and IEEE guidance, the model is a tool and is not credited as an author.

Institute for Physical AI @ BMI · The Charlot Lab
503 McKeever Rd, Arcola, TX 77583, USA
physicalai-bmi.org · contact@physicalai-bmi.org

Technical Report TR-2026-24
CC0 1.0 Universal — dedicated to the public domain.
Reference implementation: ferromotion-fluid.