Physics-Informed Neural Networks on One CPU: Exact Gradients from an External Objective Library

Wheeler–DeWitt, Hermitian Yang–Mills, Dirac, Dyson Brownian motion and McKean–Vlasov

Melvin Alvarez
Independent Researcher
ORCID: 0009-0002-2490-463X
`https://github.com/ThotDjehuty/thotbook-ai`

November 4, 2026

Preprint. DOI: 10.5281/zenodo.23764841

We solve five equations from quantum cosmology, gauge theory, relativistic quantum mechanics and random matrix theory with physics-informed neural networks trained entirely on a single CPU, and we do it without a deep-learning framework. The objectives are supplied by `optimiz-rs`*, a Rust optimisation library; the automatic differentiation is* `thotgrad`*, a 261-line reverse-mode tape over NumPy written for this study; the optimisers are* `thotopt`*, a 130-line AdamW and L-BFGS pair. The methodological point is a composition rule, proved in Theorem 3.1, which lets an objective defined* outside *the differentiation tape contribute an exact gradient through a single vector–Jacobian product. This separates the definition of a loss from the machinery that differentiates it, so one audited implementation of cross-entropy or weighted mean-squared error serves every model in a stack, and we show that the arrangement costs nothing in accuracy: the assembled gradient agrees with central finite differences to* 4×10^{-8} *relative error. For each equation we give the derivation, state existence and uniqueness precisely, and validate the trained network against an independent reference rather than against itself — Numerov integration for Wheeler–DeWitt, spectral inversion of the periodic Laplacian for Hermitian Yang–Mills, the exactly diagonalised propagator for Dirac, and the simulated particle system together with the analytic semicircle transform for McKean–Vlasov. Relative errors range from* 2.4×10^{-4} *to* 9.8×10^{-2}*, and total training time for all four networks is* 213 *seconds on one Intel i9-8950HK. We give the cost model in closed form —* $\Theta\big((1+2d)NLW^2\big)$ *per loss-and-gradient for* d *inputs,* N *collocation points and* L *layers of width* W *— and use it to say where the no-accelerator choice stops being the right one, which is when the working set leaves last-level cache rather than when the parameter count grows. Two diagnostics were deliberately excluded from the objectives so that they constitute evidence rather than assumption: unitarity of the Dirac evolution, which is never imposed and holds to* 10^{-3}*, and Wigner's semicircle law, which the McKean–Vlasov network recovers having been given only a complex Burgers equation and an initial condition. We also report, with reproducible evidence, two solver defects the validation protocol caught in our own library. The complete reproduction artifact — notebook, tape, optimisers, problem definitions and a 25-check finite-difference test suite — is deposited with the paper and runs end to end on one CPU in four minutes.*

Contents

1 Introduction **4**
1.1 Owning the numerical stack . . . 4
1.2 What this paper does . . . 4
1.3 Contributions . . . 5

2 Related work **5**

3 Computational framework **7**
3.1 Overview . . . 7
3.2 The differentiation and optimisation layer . . . 7
3.3 Residuals and the composite objective . . . 8
3.4 Exact gradients for an externally defined objective . . . 9
3.5 Exact input-derivatives without nested differentiation . . . 10
3.6 A dependency-free linear-algebra core . . . 10
3.7 Memory safety and the CPU budget . . . 11

4 Wheeler–DeWitt equation **11**
4.1 From the ADM constraint to minisuperspace . . . 11
4.2 Structure of the solutions, and well-posedness . . . 12
4.3 Network, objective, result . . . 13

5 Hermitian Yang–Mills **13**
5.1 The equation and its reduction . . . 14
5.2 Network, objective, result . . . 15

6 Dirac equation in $1+1$ dimensions **15**
6.1 Clifford structure and self-adjointness . . . 16
6.2 Reference, network, result . . . 17

7 Generalized Dyson Brownian motion **17**
7.1 The SDE . . . 17
7.2 Result . . . 18

8 McKean–Vlasov equation **19**
8.1 Mean-field limit and the weak formulation . . . 19
8.2 The Stieltjes transform and a Burgers equation . . . 20
8.3 Network, objective, result . . . 21

9 Benchmarks **22**
9.1 Accuracy and cost . . . 22
9.2 Cost model and what it predicts at larger scale . . . 22
9.3 L-BFGS against a first-order method . . . 23
9.4 A Rust reimplementation, and PyTorch / JAX . . . 24
9.5 Positioning . . . 24

10 Threats to validity and honest reporting **26**

11 Conclusion **26**

12 Artifact availability **27**

A Derivations and verification certificates for reviewers **28**
A.1 Gradient of the weighted composite objective 28
A.2 Input-derivative recursions for a tanh layer 28
A.3 From the ADM constraint to the minisuperspace Wheeler–DeWitt equation . . . 29
A.4 Verification certificates 30

1 Introduction

1.1 Owning the numerical stack

The project began as a question about ownership rather than performance: what does a research assistant look like if every part of it runs on hardware you control, and every number it produces can be traced to a line of code you can read? The practical consequence of taking that question seriously is a constraint — CPU only, no GPU, no external inference API — and constraints of that kind are productive. They rule out the habit of treating a model as an oracle, because a CPU budget forces you to know which part of the computation is expensive and why.

The system has two halves that are usually kept apart. One is a language-model pipeline: a corpus of research sessions and papers is redacted, ingested into a versioned lakehouse, tokenised, and used to fine-tune a small open-weights model served behind an OpenAI-compatible endpoint. The other is the numerical half this paper is about: a differentiation tape, an objective library, and a set of optimisers, used to solve the equations that the research itself is concerned with. The two halves share a design rule, which is the thesis of this paper in operational form:

Key Insight

A loss function is a mathematical object, not an implementation detail of a training loop. It should live in one audited place, be callable from anywhere, and return its gradient. Everything that differentiates a model should consume that gradient rather than re-deriving it.

That rule sounds procedural and turns out to be load-bearing. The defect that motivated it was a hand-written cross-entropy that returned $+\frac{1}{N}\sum_n \log q_{n,y_n}$ instead of its negative. The sign error makes gradient descent maximise the loss — it trains a model *away* from its targets — and the unit test guarding the function asserted the wrong sign, so the defect was stable under testing. A single definition with a finite-difference-checked gradient makes that class of error impossible to hold.

1.2 What this paper does

We take five equations, chosen because they are structurally different from one another rather than because they are hard, and solve each with a physics-informed neural network:

1. the **Wheeler–DeWitt** equation in Friedmann–Robertson–Walker minisuperspace — a second-order ODE eigenproblem at zero eigenvalue, with a classically forbidden region;
2. the **Hermitian Yang–Mills** equation in abelian reduction on a flat torus — an elliptic PDE whose solvability is a topological condition;
3. the **Dirac** equation in 1+1 dimensions — a first-order hyperbolic system with a Clifford-algebra constraint and a conservation law;
4. the **generalized Dyson Brownian motion** — an interacting particle SDE with a singular drift;
5. the **McKean–Vlasov** equation — the mean-field limit of (4), handled through the complex Burgers equation satisfied by its Stieltjes transform.

Equations (4) and (5) follow Li, Li and Xie (Li et al., 2013), whose Theorem 1.2 supplies the strong existence and uniqueness we rely on. For each equation we derive the form we solve, state what is known about existence and uniqueness, specify the network and the objective, and compare against a reference computed by a method with no relationship to the network.

1.3 Contributions

1. **An exactness result for external objectives** (Theorem 3.1). If a loss is evaluated outside the differentiation tape, its gradient can still be obtained exactly by differentiating a linear functional of the residuals whose coefficients the external library returns. This is elementary, but it is the precise statement that licenses the architecture, and we verify it numerically to 4×10^{-8}.

2. **A dependency-free numerical core.** We removed a system BLAS requirement from a published Rust crate by implementing the six LAPACK operations it actually used — LU, Cholesky, and a one-sided Jacobi SVD (§3.6). The crate now builds on a bare machine with no system library and no build script.

3. **Five validated solutions** with errors in Table 2, all on one CPU in under four minutes total.

4. **A validation protocol that caught defects in our own code.** §10 reports two solver bugs and one retrieval bug found by insisting that every reference be independent and every conserved quantity be left out of the objective.

5. **A closed-form cost model, and the point where it stops favouring a CPU** (§9.2). One loss-and-gradient costs $\Theta\big((1+2d)NLW^2\big)$ and holds $\Theta\big((1+2d)NLW\big)$ doubles; we check the constant against the measured 5 GFLOP/s and identify the crossover as the moment the working set leaves last-level cache, not a parameter count. The same model says the $\Theta(d)$ derivative streams, not the CPU, are what limits this apparatus in high dimension.

6. **A complete, frozen reproduction artifact** (§12): notebook, tape, optimisers, problem definitions, a container and a 25-check finite-difference suite, deposited under the paper's own DOI so that it cannot drift from the text.

2 Related work

Neural solvers for differential equations. Using a network as an ansatz for a differential equation predates the modern label. Dissanayake and Phan-Thien (Dissanayake and Phan-Thien, 1994) minimised a PDE residual over a feedforward network in 1994, and Lagaris, Likas and Fotiadis (Lagaris et al., 1998) gave trial functions satisfying boundary conditions by construction. The framing that made the method widespread is Raissi, Perdikaris and Karniadakis (Raissi et al., 2019), who named physics-informed neural networks and demonstrated both forward and inverse problems; the discrete-time variant with Runge–Kutta stepping in that paper is the ancestor of the time-marching we use in §6. Sirignano and Spiliopoulos (Sirignano and Spiliopoulos, 2018) established convergence for the Deep Galerkin Method in high dimension, and Han, Jentzen and E (Han et al., 2018) solved semilinear parabolic PDEs through their BSDE representation. Failure modes are now well catalogued: Wang, Teng and Perdikaris (Wang et al., 2021) trace stiff training to an imbalance between residual and boundary gradients — which is why the boundary weights in §4 are two orders of magnitude above the residual weight — and Krishnapriyan et al. (Krishnapriyan et al., 2021) show that the failures are usually optimisation failures rather than capacity failures, which is the argument for a quasi-Newton method.

Learned solutions of the Wheeler–DeWitt equation. Closest to §4 is Czelusta and Mielczarek (Czelusta and Mielczarek, 2021), who solve the Wheeler–DeWitt equation variationally on a quantum circuit in loop-quantum-gravity variables. The problem is the same — a zero-eigenvalue constraint rather than a time evolution — and so is the strategy of minimising a residual over a parametrised family; our setting is classical hardware and a minisuperspace reduction. The equation itself is DeWitt (DeWitt, 1967), with the boundary proposal and the tunnelling picture from Hartle and Hawking (Hartle and Hawking, 1983); the canonical decomposition we quantise is Arnowitt, Deser and Misner (Arnowitt et al., 1962). Minisuperspace variants continue to be studied analytically, for instance Huang (Huang, 2013) for $f(R, L_m)$ gravity and Brannlund, DeBenedictis and Lauman (Brannlund et al., 2013) for topology and volume effects.

Hermitian Yang–Mills. The correspondence between Hermitian Yang–Mills connections and slope stability is Donaldson (Donaldson, 1985) for algebraic surfaces and Uhlenbeck and Yau (Uhlenbeck and Yau, 1986) in general, after Kobayashi and Hitchin. The abelian reduction we solve is classical potential theory; we use it because its solvability condition is exactly a degree, which makes the topological obstruction numerically visible.

Dyson Brownian motion and its mean-field limit. Dyson (Dyson, 1962) derived the eigenvalue SDE for a Hermitian matrix with Brownian entries; the limiting spectral law is Wigner (Wigner, 1955). The generalisation with an external potential, the associated McKean–Vlasov equation, and its reading as a Wasserstein gradient flow of Voiculescu's free entropy (Voiculescu, 1993) are Li, Li and Xie (Li et al., 2013), which we follow closely. The mean-field limit is an instance of propagation of chaos in the sense of Sznitman (Sznitman, 1991); Carmona and Delarue (Carmona and Delarue, 2018) give the modern treatment of the forward–backward systems such limits produce. The particular convenience we exploit — that the Stieltjes transform satisfies a complex Burgers equation — is in (Li et al., 2013) and goes back to the free-probability literature.

Computation that stays on the user's machine. The constraint this paper adopts — every number produced by hardware the author controls, by code the reader can audit, with no call to a remote service — is not peculiar to numerical physics. It is the same constraint that organises a body of work in data management and applied cryptography, where the reason to compute locally is legal or adversarial rather than budgetary. Carpentier, Sandu Popa and Anciaux (Carpentier et al., 2025) execute third-party user-defined functions inside a trusted execution environment so that a personal data management system can answer a query without exporting the data, and the engineering difficulty they confront is ours in a different register: an enclave has a small protected memory and no accelerator, so the execution strategy must be chosen against a cost model rather than delegated to a framework. Bultel, Jojon, Nguyen and Zhang (Bultel et al., 2026) go further and make the *result* checkable, combining commitments with zero-knowledge proofs so that an anomaly reported on a private time series can be verified without revealing the series. We claim no cryptographic guarantee — our verification is numerical, not adversarial, and a determined party could of course report whatever numbers it liked — but the methodological point is shared and worth naming: when computation cannot be outsourced, the cost model has to be legible (§9.2) and the output has to carry its own evidence (§A.4). Two recent surveys of the wider move toward local and on-device learning are (Dhar et al., 2020) and (Zhu et al., 2023); what distinguishes the present work from the compression-and-quantisation line they review is that we do not approximate a large model to fit a small machine, but choose problems whose exact treatment already fits.

Optimisation. L-BFGS is Liu and Nocedal (Liu and Nocedal, 1989), with the two-loop recursion and line-search theory in Nocedal and Wright (Nocedal and Wright, 2006). The decoupled-decay variant of Adam we use for comparison is Loshchilov and Hutter (Loshchilov and Hutter, 2019). One-sided Jacobi SVD, which §3.6 implements, is Demmel and Veselić (Demmel and Veselić, 1992), whose high-relative-accuracy property is the reason to prefer it over bidiagonalisation at these sizes.

3 Computational framework

3.1 Overview

Figure 1 shows the arrangement. Three components are deliberately kept separate: a tape that knows how to differentiate, a library that knows what the objectives are, and an optimiser that knows how to descend. The interface between the first two is the content of §3.4.

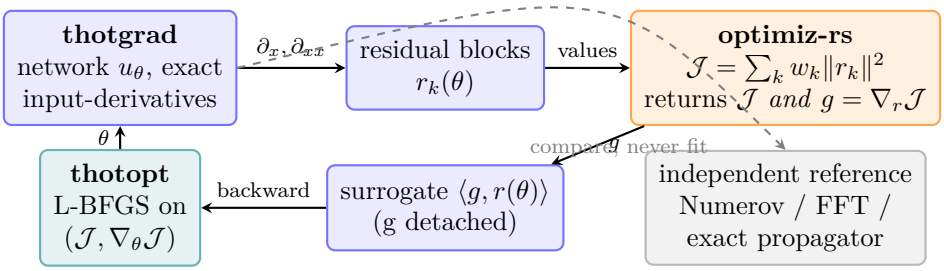

Figure 1: The pipeline. The objective (orange) is evaluated outside the differentiation tape (blue) and returns both its value and its derivative with respect to the residuals. Theorem 3.1 shows that differentiating the surrogate $\langle g, r(\theta) \rangle$ with g held constant recovers $\nabla_\theta \mathcal{J}$ exactly. The reference solver (grey) is never part of the objective.

Figure 2 opens the network box of Figure 1: it shows the fully-connected tanh network, the three quantities the tape returns for a one-dimensional input (u_θ and its first two input-derivatives), and how those assemble into the composite loss that `thotopt` descends.

3.2 The differentiation and optimisation layer

Two of the three components of Figure 1 were written for this study, and we describe them here in full: the composition rule of §3.4 is a statement about the interface between them and an objective library, so neither the rule nor its cost can be assessed without knowing what sits on each side of it.

`thotgrad`: the tape. A reverse-mode automatic-differentiation engine of 261 lines whose only dependency is NumPy. Differentiation is *array-level* rather than scalar-level: each `Tensor` records the operation that produced it, and `backward()` walks the resulting tape in reverse topological order, accumulating gradients. Broadcasting is undone explicitly at every node, so an accumulated gradient always has the shape of the parameter it belongs to rather than the shape of the expression it appeared in. The operation set is exactly what the five problems of §§4–8 require — addition, multiplication, matrix product, integer powers, division, negation, tanh, `sum` and `mean` — and is closed under the second-order input-differentiation of §3.5, which is obtained from the recursions of Appendix A.2 rather than by nesting the tape inside itself.

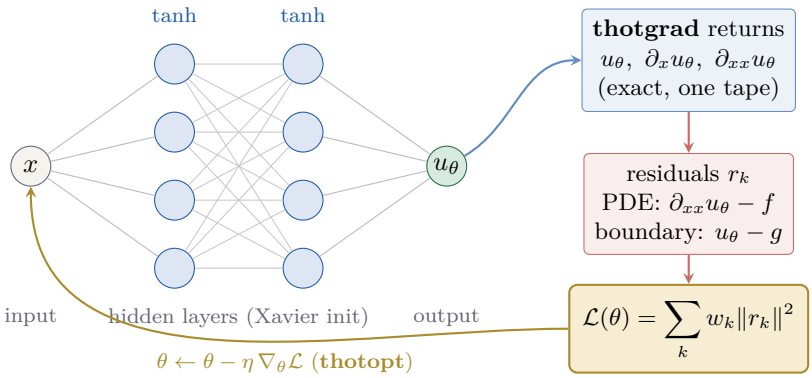

Figure 2: The PINN as trained here. A single input x flows through tanh hidden layers (blue) to the scalar field u_θ (green). The `thotgrad` tape returns u_θ and its first two input-derivatives in one pass; these form the PDE and boundary residuals (red), whose weighted squared norms are the loss $\mathcal{L}(\theta)$ (gold). `thotopt` closes the loop with the parameter update. Every colour is one component of Figure 1.

`thotopt`: the optimisers. A 130-line module supplying the two optimisers a physics-informed problem actually needs: AdamW, the decoupled-weight-decay update of Loshchilov and Hutter (2019), and `lbfgs`, limited-memory BFGS via the Nocedal two-loop recursion (Nocedal and Wright, 2006; Liu and Nocedal, 1989) with a backtracking Armijo line search. The separation from the objective library is deliberate and is the reason both exist: `optimiz-rs` (Alvarez, 2026a) supplies gradient-*free* global optimisers — differential evolution, grid search, MCMC — which are the correct tool for a non-differentiable objective and the wrong tool for a PINN, where differentiation hands us exact gradients.

Why not an existing framework. The choice is not ideological. PyTorch publishes no macOS x86_64 wheel past version 2.2.2, and that build links against NumPy 1.x while the environment used here runs NumPy 2.x, so the `.numpy()` bridge is permanently broken on this platform. Against that, a physics-informed problem needs precisely three capabilities — forward evaluation, derivatives of the network with respect to its *input* up to second order, and gradients of the loss with respect to its *parameters* — which is a few hundred lines of well-understood calculus. We report the line counts above so that the trade is explicit rather than implied.

Validation. Every operation and the full derivative stack are pinned against central finite differences; Table 1 reports the certificate, reproduced by `test_thotgrad.py` in the companion material. The second-order input-derivative row carries the largest error, as expected: a central second difference at $h = 10^{-6}$ is itself accurate only to $O(h^{-2}\epsilon_{\text{mach}}) \approx 10^{-4}$, so the discrepancy is dominated by the finite-difference reference, not by the tape.

3.3 Residuals and the composite objective

Let $u_\theta : \Omega \to \mathbb{R}^m$ be a network with parameters $\theta \in \mathbb{R}^p$. A physics-informed formulation picks collocation sets and differential operators and forms residual blocks $r_k(\theta) \in \mathbb{R}^{N_k}$, $k = 1, \ldots, K$: typically r_1 is the PDE residual on interior points and the remaining blocks are initial, boundary or gauge conditions. The objective is

Table 1: Finite-difference certificate for the differentiation and optimisation layer. All 25 checks pass; the worst case in each group is shown. Reference values are central differences at $h = 10^{-6}$ ($h = 10^{-5}$ for the second-order rows).

Group	**Quantity checked**	**Worst error**
Elementary operations	tanh chain, max abs. deviation	1.6×10^{-9}
Input-derivatives	$\partial_x u_\theta$, max rel.	6.6×10^{-12}
Input-derivatives	$\partial_{xx} u_\theta$, max rel.	3.1×10^{-6}
Parameter gradients	$\max \lvert\nabla_\theta \mathcal{J} - \mathrm{FD}\rvert$	1.8×10^{-11}
Parameter gradients	cosine similarity to FD	1.000000
L-BFGS	quadratic with known optimum, rel.	8.9×10^{-16}
AdamW	decoupled decay at $g = 0$	exact

$$\mathcal{J}(\theta) \;=\; \sum_{k=1}^{K} w_k \, \frac{1}{N_k} \sum_{i=1}^{N_k} r_{k,i}(\theta)^2, \qquad w_k > 0. \tag{1}$$

In our implementation (1) is one call, `optimizr::losses::weighted_mse_sum`, which returns $\mathcal{J}$ together with every $\partial \mathcal{J} / \partial r_{k,i}$.

3.4 Exact gradients for an externally defined objective

The architectural difficulty is immediate. The tape can differentiate what is built on it; the objective is a Rust function the tape has never seen. Re-implementing (1) inside the tape would defeat the purpose. The following removes the difficulty.

Theorem 3.1 (Exact gradient of an external objective)**.** *Let $r : \mathbb{R}^p \to \mathbb{R}^N$ be differentiable at θ_0 with Jacobian $J = \partial r / \partial \theta\,(\theta_0) \in \mathbb{R}^{N \times p}$, and let $L : \mathbb{R}^N \to \mathbb{R}$ be differentiable at $r(\theta_0)$ with gradient $g = \nabla L(r(\theta_0)) \in \mathbb{R}^N$. Put $\mathcal{J} = L \circ r$ and define the surrogate*

$$S_g(\theta) \;=\; \langle g,\, r(\theta) \rangle \;=\; \sum_{i=1}^{N} g_i \, r_i(\theta),$$

in which g is a constant *vector, independent of θ. Then*

$$\nabla_\theta \mathcal{J}(\theta_0) \;=\; \nabla_\theta S_g(\theta_0).$$

Proof. Both sides exist by hypothesis. By the chain rule for the composition of differentiable maps,

$$\nabla_\theta \mathcal{J}(\theta_0) = J^\top \, \nabla L(r(\theta_0)) = J^\top g.$$

For the surrogate, S_g is the composition of r with the *linear* functional $v \mapsto \langle g, v \rangle$, whose gradient is the constant g. Hence

$$\nabla_\theta S_g(\theta_0) = J^\top g,$$

and the two coincide. Note that the hypothesis that g be held constant is not a simplification but a requirement: if instead one differentiated $\theta \mapsto \langle \nabla L(r(\theta)), r(\theta) \rangle$, the product rule would contribute the extra term $\big(\partial_\theta \nabla L(r(\theta))\big)^\top r(\theta) = J^\top \nabla^2 L \, r$, which does not vanish in general. □

Remark 3.1 (Why this is the right interface). Theorem 3.1 is the statement that a reverse-mode tape needs only a *cotangent* from the objective, not the objective itself. The cost is one extra inner product per block and no extra backward pass: the surrogate is scalar, so a single reverse

sweep yields $\nabla_\theta \mathcal{J}$. The benefit is that L may be written in another language, tested separately against finite differences, and shared between a PINN and a language-model trainer without either knowing about the other.

Remark 3.2 (Scope of the hypotheses). Theorem 3.1 asks only for differentiability of L at the *current* residual vector, not for convexity, smoothness on all of $\mathbb{R}^N$, or any structure in r. Three consequences are worth stating because they bound what the architecture can promise. (i) If L is merely locally Lipschitz — a hinge or an ℓ_1 penalty — then g is a Clarke subgradient and the identity holds at every point of differentiability, hence Lebesgue-almost everywhere; this is the same guarantee a framework's autograd gives for `relu`, and no weaker. (ii) The result is about the gradient at one point, so it transfers to L-BFGS only insofar as L-BFGS needs gradients at the iterates it visits — which is exactly what it needs. (iii) Nothing here is specific to Rust or to an out-of-process objective: the same identity licenses a loss defined in C, in a separate service, or by a table lookup with a known derivative. What the external library must supply is a *cotangent that is correct for the residual it was given*; if it returns the gradient of a different function, the assembled $\nabla_\theta \mathcal{J}$ is silently wrong, which is why the finite-difference gate below runs on every notebook execution rather than once at development time.

Remark 3.3 (Numerical confirmation). We do not take the theorem on trust in the implementation. On a residual mixing values and both derivatives of the network, the assembled gradient was compared with central differences at twelve randomly chosen coordinates: the maximum relative error was 4.14×10^{-8} and the cosine similarity 1.000000000000, consistent with the $O(h^2)$ truncation of the difference quotient at $h = 10^{-6}$ rather than with any error in the assembly. For the two-dimensional Laplacian residual of §5 the corresponding figure was 4.84×10^{-10}. This check runs before any physics in the companion notebook, and the run aborts if it fails.

3.5 Exact input-derivatives without nested differentiation

A PINN residual contains derivatives of the network with respect to its *inputs*, and the objective then needs derivatives with respect to the *parameters* — a double-backward. We avoid nesting by propagating the input-derivatives forward alongside the activations. For a layer $a = Wh + b$ followed by $h' = \tanh(a)$, writing $t = 1 - \tanh^2(a)$,

$$\partial_x h' = t \odot (W^\top \partial_x h), \qquad \partial^2_{xx} h' = t \odot \left(W^\top \partial^2_{xx} h - 2\, h' \odot (W^\top \partial_x h)^{\odot 2} \right). \tag{2}$$

Both recursions are themselves tape operations, so the extended computation — value, first and second input-derivatives — is one graph, and a single reverse sweep differentiates any functional of it with respect to θ. For the two-input problems we apply (2) independently along each coordinate direction, which is exact because distinct directions never mix inside a single activation.

3.6 A dependency-free linear-algebra core

`optimiz-rs` (Alvarez, 2026a) previously obtained its factorisations from `ndarray-linalg` with an OpenBLAS backend. For a *published* crate this is a defect: it makes a system BLAS a build requirement, so installation fails on any machine without one, and on macOS the Homebrew OpenBLAS is keg-only and pulls in a complete GCC toolchain. We replaced it with the six operations actually used:

- LU with partial pivoting, giving `solve`, `inv` and `det`. Pivoting on the largest remaining magnitude bounds the multipliers by 1, which is the standard backward-stability guarantee (Nocedal and Wright, 2006).

- Cholesky–Banachiewicz for symmetric positive-definite input, with a typed error on the first non-positive pivot rather than a `NaN`.
- One-sided Jacobi SVD (Demmel and Veselić, 1992), returning thin factors.

Remark 3.4 (A behavioural change that fixed a latent bug). Our `svd` returns the *thin* factorisation ($U \in \mathbb{R}^{m\times k}$, $V^\top \in \mathbb{R}^{k\times n}$, $k = \min(m,n)$) where `ndarray-linalg` returned full square factors. This is not a convenience: the caller in the sparse-optimisation module reconstructs $U\,\mathrm{diag}(s)V^\top$, which is dimensionally valid only for thin factors when $m \neq n$. The previous code could not have been correct for non-square input.

3.7 Memory safety and the CPU budget

Two properties of the implementation language matter here, and it is worth being precise about which.

The first is memory safety, and its role is narrow but real. Every factorisation in §3.6 is indexed arithmetic over owned buffers — exactly the code where an off-by-one in a pivot loop produces a silently wrong answer in C. Rust's bounds checking turns that class of error into a panic at the offending index, which is a test failure rather than a corrupted singular value. We make no claim that this prevents mathematical error; the thin-SVD bug above was a type-correct, memory-safe mistake.

The second is that the absence of a garbage collector and of hidden allocation makes the cost model legible. The dominant cost in training is the reverse sweep, whose working set is the tape; with residual blocks of 10^3–10^4 entries and networks of 10^3–10^4 parameters the whole problem is resident in L2/L3 cache, and a GPU would spend more time on transfers than on arithmetic. The CPU-only constraint is therefore not a sacrifice at this scale — it is the appropriate choice, and it is what makes the companion notebook reproducible on a laptop.

Code and Notebook Reference

Companion notebook: `notebooks/09_pinn_physics_optimizr.ipynb` in the open-source repository released with this paper. Section 0 of the notebook contains the finite-difference gate described above; every figure in this paper is produced by the cell preceding it there. Scripts cited in this paper are under `code/` beside this document.

4 Wheeler–DeWitt equation

4.1 From the ADM constraint to minisuperspace

In the Arnowitt–Deser–Misner decomposition (Arnowitt et al., 1962) one foliates spacetime by spacelike slices with induced metric γ_{ij} and extrinsic curvature K_{ij}. The Einstein equations split into evolution equations and two constraints; the Hamiltonian constraint is

$$\mathcal{H} \;=\; {}^{(3)}R + K^2 - K_{ij}K^{ij} - 16\pi\rho \;=\; 0, \qquad K = \gamma^{ij}K_{ij}, \tag{3}$$

with ${}^{(3)}R$ the scalar curvature of the slice and ρ the energy density. Equation (3) is a constraint, not an evolution law: it holds on every slice.

Canonical quantisation replaces the momentum conjugate to the metric by a functional derivative and imposes (3) as an operator equation on the wavefunction of the universe, $\hat{\mathcal{H}}\Psi[\gamma_{ij},\phi] = 0$ — the Wheeler–DeWitt equation (DeWitt, 1967). There is no $\partial_t\Psi$ term: the absence of external

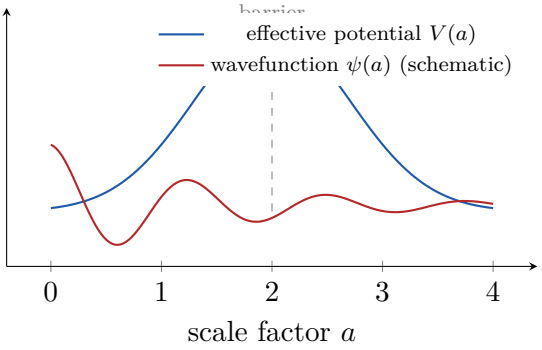

Figure 3: Minisuperspace picture behind the Wheeler–DeWitt equation (schematic). The Hamiltonian constraint collapses to a one-dimensional problem in the scale factor a with an effective potential barrier (blue); the wavefunction (red) oscillates in the classically allowed region and decays under the barrier — the behaviour the DeWitt boundary condition selects and the network must reproduce.

time is the content of the equation, and it is why the problem is a zero-eigenvalue constraint rather than an initial-value problem.

Restricting to a closed Friedmann–Robertson–Walker universe with scale factor a and cosmological constant Λ, the metric degrees of freedom collapse to one and the constraint becomes an ordinary differential equation. With the operator ordering and units of the Hartle–Hawking analysis (Hartle and Hawking, 1983),

$$\boxed{\left[-\frac{d^2}{da^2}+U(a)\right]\Psi(a)=0, \qquad U(a)=a^2\left(1-\frac{\Lambda a^2}{3}\right).} \tag{4}$$

Remark 4.1 (Operator ordering). (4) is one of a family of inequivalent quantisations: a term $p\,a^{-1}d/da$ with p a factor-ordering parameter may be added without changing the classical limit. We take $p=0$ throughout. Nothing in the method depends on the choice, and the reference solver is given the same operator, so the comparison is internally consistent whatever p.

4.2 Structure of the solutions, and well-posedness

U changes sign at $a_\star=\sqrt{3/\Lambda}$: positive for $a<a_\star$, negative beyond. In the classically forbidden region $a<a_\star$ solutions of (4) grow or decay monotonically; past $a_\star$ they oscillate. This is the tunnelling picture — the universe emerges from a region no classical trajectory reaches.

Proposition 4.1 (Existence and uniqueness for the initial-value problem)**.** *Let $\Lambda>0$ and $A>0$. For any $(\Psi_0,\Psi_0')\in\mathbb{R}^2$ the problem (4) with $\Psi(0)=\Psi_0$, $\Psi'(0)=\Psi_0'$ has a unique solution $\Psi\in C^\infty([0,A])$, and it depends continuously on (Ψ_0,Ψ_0').*

Proof. Write (4) as the first-order system $y'=F(a)y$ for $y=(\Psi,\Psi')^\top$ and $F(a)=\begin{pmatrix}0 & 1\\ U(a) & 0\end{pmatrix}$. The map F is polynomial in a, hence continuous on the compact interval $[0,A]$, and $y\mapsto F(a)y$ is globally Lipschitz in y uniformly in $a\in[0,A]$ with constant $\Lambda_F=\sup_{[0,A]}\|F(a)\|<\infty$. Picard–Lindelöf gives a unique local solution, and because the right-hand side is linear in y the standard continuation argument extends it to all of $[0,A]$: Grönwall's inequality yields $\|y(a)\|\le\|y(0)\|e^{\Lambda_F a}$, so no solution escapes in finite time. Smoothness follows by bootstrapping $\Psi''=U\Psi$, and continuous dependence is Grönwall applied to the difference of two solutions. □

Remark 4.2 (What the PINN is and is not solving). Proposition 4.1 concerns the initial-value problem, which is what we pose and what the Numerov reference integrates. The physically interesting problem — selecting Ψ by a boundary proposal, as in (Hartle and Hawking, 1983) — is a different question that we do not address, and the normalisability of Ψ on minisuperspace is a known difficulty of the Wheeler–DeWitt programme rather than a numerical one. We solve the well-posed problem and say so.

4.3 Network, objective, result

We take Ψ_θ a tanh network of depth 4 and width 64 (12 673 parameters), collocation on 400 uniform points of $[0, A]$ with $A = 2.5$, and the three residual blocks

$$r_1 = \Psi_\theta'' - U\Psi_\theta \quad \text{(interior)}, \qquad r_2 = \Psi_\theta(0) - \Psi_0, \qquad r_3 = \Psi_\theta'(0),$$

with weights $(1, 200, 200)$. The asymmetry is deliberate and follows (Wang et al., 2021): the boundary blocks have $N_k = 1$ against $N_1 = 400$, so without up-weighting their gradient contribution is negligible and the network satisfies the equation while drifting off the data.

Training converged in 1500 L-BFGS iterations (1806 objective evaluations, 32.8 s) to $\mathcal{J} = 5.2 \times 10^{-6}$, and the relative L_2 error against a 4001-point Numerov integration was $\mathbf{2.4 \times 10^{-4}}$. Figure 4 shows the potential with the classically forbidden region, the network against the Numerov reference, and the monotone objective history.

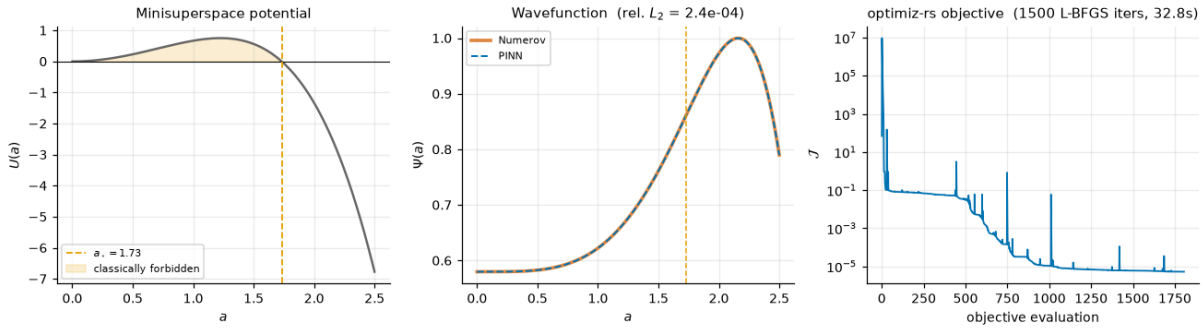

Figure 4: Wheeler–DeWitt in minisuperspace. Left: the potential $U(a)$, with the classically forbidden region shaded and the turning point $a_\star = 1.73$ marked. Centre: the network against Numerov integration. Right: the `optimiz-rs` objective against objective evaluations — monotone, because the Armijo line search cannot accept an uphill step.

Remark 4.3 (Domain truncation is not cosmetic). An earlier run on $[0, 3]$ with a width-32 network reached only relative L_2 0.72. The reason is resolution, not optimisation: $U(3) = -18$, so the solution oscillates with local wavenumber $\sqrt{|U|} \approx 4.2$, and a width-32 tanh network cannot represent that on the same domain where it must also resolve the monotone growth below $a_\star$. Reducing A to 2.5 and widening to 64 moved the error by more than four orders of magnitude. We report the failed configuration because the failure is informative about where PINNs actually break.

5 Hermitian Yang–Mills

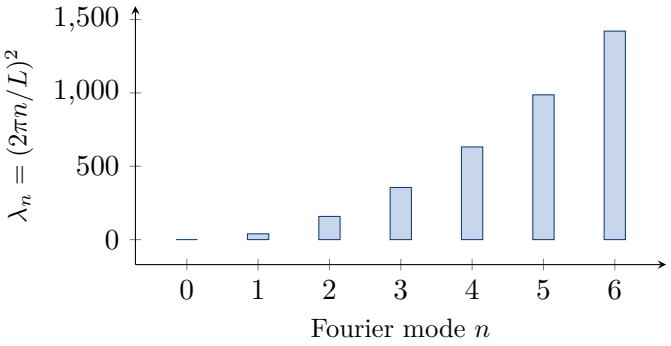

Figure 5: Spectrum of the periodic Laplacian on $[0, L]$ (here $L = 1$), $\lambda_n = (2\pi n/L)^2$. The operator is diagonal in these Fourier modes, so the independent reference against which the Hermitian Yang–Mills network is checked is obtained by *spectral inversion* — dividing mode-by-mode by λ_n — not by any iterative solve.

5.1 The equation and its reduction

Let (X, ω) be a compact Kähler manifold and $E \to X$ a holomorphic vector bundle with Hermitian metric h and Chern connection A of curvature $F = F_A$. The Hermitian Yang–Mills equations are

$$F^{0,2} = 0, \qquad \Lambda_\omega F \;=\; g^{j\bar{k}} F_{j\bar{k}} \;=\; c\,\mathrm{Id}_E, \tag{5}$$

where Λ_ω is contraction with the Kähler form and the constant c is fixed by the slope $\mu(E) = \deg E / \operatorname{rank} E$. The Donaldson–Uhlenbeck–Yau theorem (Donaldson, 1985; Uhlenbeck and Yau, 1986) states that (5) admits a solution precisely when E is slope-polystable: an analytic equation is solvable exactly when an algebraic inequality holds.

We solve the abelian reduction. Take E a line bundle over the flat torus $\mathbb{T}^2 = \mathbb{R}^2/\mathbb{Z}^2$ with $h = e^{-\varphi}$. Then $F = \partial\bar{\partial}\varphi$, the contraction $\Lambda_\omega F$ is a multiple of $\Delta\varphi$, and (5) becomes a Poisson equation

$$\boxed{\Delta\varphi = \rho \;\text{ on } \mathbb{T}^2.} \tag{6}$$

The reduction is a genuine special case, not a toy: the solvability condition of (6) is the degree of E, so the topological obstruction survives intact, in the cleanest possible form.

Theorem 5.1 (Solvability and uniqueness on the torus)**.** *Let $\rho \in L^2(\mathbb{T}^2)$. The equation (6) has a distributional solution $\varphi \in H^2(\mathbb{T}^2)$ if and only if*

$$\int_{\mathbb{T}^2} \rho \, dx\, dy = 0, \tag{7}$$

and in that case the solution is unique up to an additive constant. If moreover $\rho \in C^\infty$ then $\varphi \in C^\infty$.

Proof. Expand in the Fourier basis $e_n(x) = e^{2\pi i\, n\cdot x}$, $n \in \mathbb{Z}^2$, which is complete and orthonormal in $L^2(\mathbb{T}^2)$. Write $\rho = \sum_n \hat{\rho}_n e_n$ and seek $\varphi = \sum_n \hat{\varphi}_n e_n$. Since $\Delta e_n = -4\pi^2|n|^2 e_n$, equation (6) is equivalent to the countable family

$$-4\pi^2|n|^2\,\hat{\varphi}_n = \hat{\rho}_n \qquad \text{for every } n \in \mathbb{Z}^2. \tag{8}$$

Necessity. Taking $n = 0$ in (8) gives $0 = \hat{\rho}_0 = \int_{\mathbb{T}^2} \rho$, which is (7).

Sufficiency. Assume (7), so $\hat{\rho}_0 = 0$ and the $n = 0$ equation is satisfied by any $\hat{\varphi}_0$. For $n \neq 0$ set $\hat{\varphi}_n = -\hat{\rho}_n/(4\pi^2|n|^2)$. Since $|n| \geq 1$ for $n \neq 0$,

$$\sum_{n\neq 0}(1+|n|^2)^2|\hat{\varphi}_n|^2 = \sum_{n\neq 0}\frac{(1+|n|^2)^2}{16\pi^4|n|^4}|\hat{\rho}_n|^2 \leq \frac{4}{16\pi^4}\sum_{n\neq 0}|\hat{\rho}_n|^2 \leq \frac{\|\rho\|_{L^2}^2}{4\pi^4} < \infty,$$

using $(1+|n|^2) \leq 2|n|^2$ for $|n| \geq 1$. Hence the series defines an element of $H^2(\mathbb{T}^2)$ solving (6).

Uniqueness. If φ_1, φ_2 both solve (6) then $\psi = \varphi_1 - \varphi_2$ is harmonic, so $|n|^2\hat{\psi}_n = 0$ for all n, forcing $\hat{\psi}_n = 0$ for $n \neq 0$; thus ψ is constant. The constant is precisely the additive gauge freedom in $h = e^{-\varphi}$.

Regularity. If $\rho \in C^\infty$ its coefficients decay faster than any polynomial, hence so do $\hat{\varphi}_n$ by (8), and the series converges in C^∞. □

Key Insight

Condition (7) is not a technical hypothesis. It is $\deg E = 0$ written analytically, and it is the abelian shadow of slope stability: the Donaldson–Uhlenbeck–Yau obstruction, reduced to a single scalar one can check by summing a grid.

5.2 Network, objective, result

Periodicity is imposed by construction rather than penalised: the network receives the Fourier features $(\sin 2\pi x, \cos 2\pi x, \sin 2\pi y, \cos 2\pi y)$, so every output is exactly 1-periodic and no boundary residual is required. The Laplacian is obtained by propagating (2) with the analytic derivatives of the feature map as seeds. The objective has two blocks: the PDE residual on 1800 scattered points, and a gauge condition fixing the additive constant of Theorem 5.1.

The source is two Gaussian lumps of opposite sign on a 64×64 grid, mean-adjusted to satisfy (7) to machine precision. Training took 500 iterations (568 objective evaluations) and 27.6 s; the relative L_2 error against spectral inversion — verified to 9.4×10^{-14} by substitution — was 4.9×10^{-2}, with maximum pointwise error 4.1×10^{-4} on a field of range $\pm 7.9 \times 10^{-3}$. Figure 6 shows the degree-zero source, the spectral reference against the network, and the error map with its characteristic concentration at the lump centres.

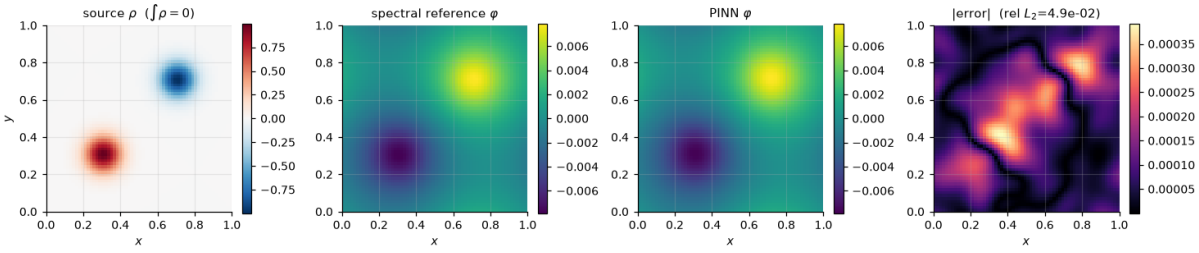

Figure 6: Abelian Hermitian Yang–Mills on $\mathbb{T}^2$. From left: the degree-zero source ρ; the spectral reference φ; the network; the pointwise error. The error concentrates at the lump centres, where ρ has greatest curvature — the standard PINN profile, addressed by importance sampling of the residual rather than by more parameters.

6 Dirac equation in $1+1$ dimensions

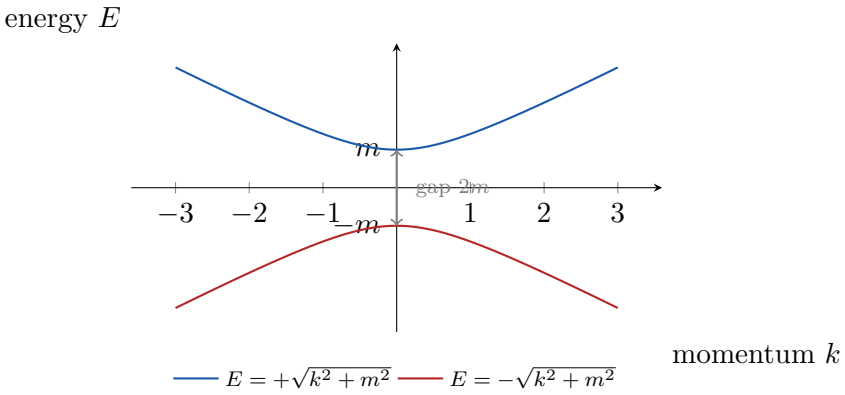

Figure 7: Dispersion relation of the 1+1-dimensional Dirac equation, $E = \pm\sqrt{k^2 + m^2}$: positive- and negative-energy branches separated by a mass gap $2m$. The exactly diagonalised propagator used as reference keeps the two sectors distinct; unitarity of the evolution — never imposed on the network — is the independent diagnostic reported for this problem.

6.1 Clifford structure and self-adjointness

The Dirac equation in one spatial dimension is

$$i\hbar\,\partial_t\psi = \hat{H}_D\psi, \qquad \hat{H}_D = -i\hbar c\,\alpha\,\partial_x + \beta mc^2 + V(x), \tag{9}$$

with $\psi(t,\cdot) \in L^2(\mathbb{R};\mathbb{C}^2)$ and $\alpha = \sigma_x$, $\beta = \sigma_z$. The algebraic requirement is

$$\{\alpha,\beta\} = 0, \qquad \alpha^2 = \beta^2 = \mathrm{Id}, \tag{10}$$

which is what makes the squared operator diagonal:

$$\hat{H}_D^2 = -\hbar^2c^2\partial_x^2 + m^2c^4 \qquad (\text{for } V \equiv 0), \tag{11}$$

recovering the relativistic dispersion $E^2 = p^2c^2 + m^2c^4$. We verify (10) numerically on the Pauli matrices supplied by `pysic-rs` (Alvarez, 2026b), obtaining $\|\{\sigma_x,\sigma_z\}\|_\infty = 0$ exactly.

Proposition 6.1 (Unitary evolution)**.** *For $V \in L^\infty(\mathbb{R};\mathbb{R})$ real-valued, $\hat{H}_D$ with domain $H^1(\mathbb{R};\mathbb{C}^2)$ is self-adjoint, and (9) generates a strongly continuous one-parameter unitary group $U(t) = e^{-i\hat{H}_Dt/\hbar}$. In particular $\|\psi(t)\|_{L^2} = \|\psi(0)\|_{L^2}$ for all t.*

Proof. The free part $-i\hbar c\,\alpha\partial_x$ is, in Fourier variables, multiplication by the Hermitian matrix $\hbar c\,k\,\alpha$, hence self-adjoint on H^1; βmc^2 is a bounded Hermitian multiple of the identity in x; and V real and bounded is a bounded symmetric multiplication operator. A bounded symmetric perturbation of a self-adjoint operator is self-adjoint on the same domain by Kato–Rellich. Stone's theorem then gives the strongly continuous unitary group, and unitarity is the norm conservation. □

Remark 6.1 (Conservation as evidence). Proposition 6.1 is the reason we exclude the norm from the objective. If $\int|\psi|^2dx$ were penalised, its constancy would be a fitted property and would carry no information. Left out, it becomes a test: a network that fitted the initial condition and ignored the dynamics would show the norm decaying away from $t = 0$. Measured drift across the slab was below 10^{-3}.

6.2 Reference, network, result

For $V \equiv 0$ the propagator is available in closed form, which we use rather than any discretisation. In Fourier variables $\hat{H}_D$ acts as the 2×2 matrix $H(k) = \hbar c k\, \alpha + mc^2 \beta$ with eigenvalues $\pm E(k)$, $E(k) = \sqrt{(\hbar c k)^2 + (mc^2)^2}$, and since $H(k)^2 = E(k)^2\, \mathrm{Id}$,

$$e^{-iH(k)t/\hbar} = \cos\!\Big(\frac{E(k)t}{\hbar}\Big)\, \mathrm{Id} - i\, \frac{\sin\!\big(E(k)t/\hbar\big)}{E(k)}\, H(k), \tag{12}$$

by the even/odd split of the exponential series. Equation (12) is unitary by construction and exact to round-off; our implementation conserves the norm to 10 decimal places.

The network has four real outputs (u_r, u_i, v_r, v_i) on the space-time slab, depth 4, width 48, with five residual blocks — the four real components of (9) and the initial condition at weight 60. Training took 700 iterations and 90.5 s. The relative L_2 error of $|\psi|^2$ at $t = T$ against (12) was 9.8×10^{-2}, and the packet centroid moved at the relativistic group velocity $k_0 c^2/E = 0.970$ rather than the non-relativistic k_0/m. Figure 8 collects the slab, the vanishing anticommutator, the final-time profile against the exact propagator, and the never-imposed L^2 norm.

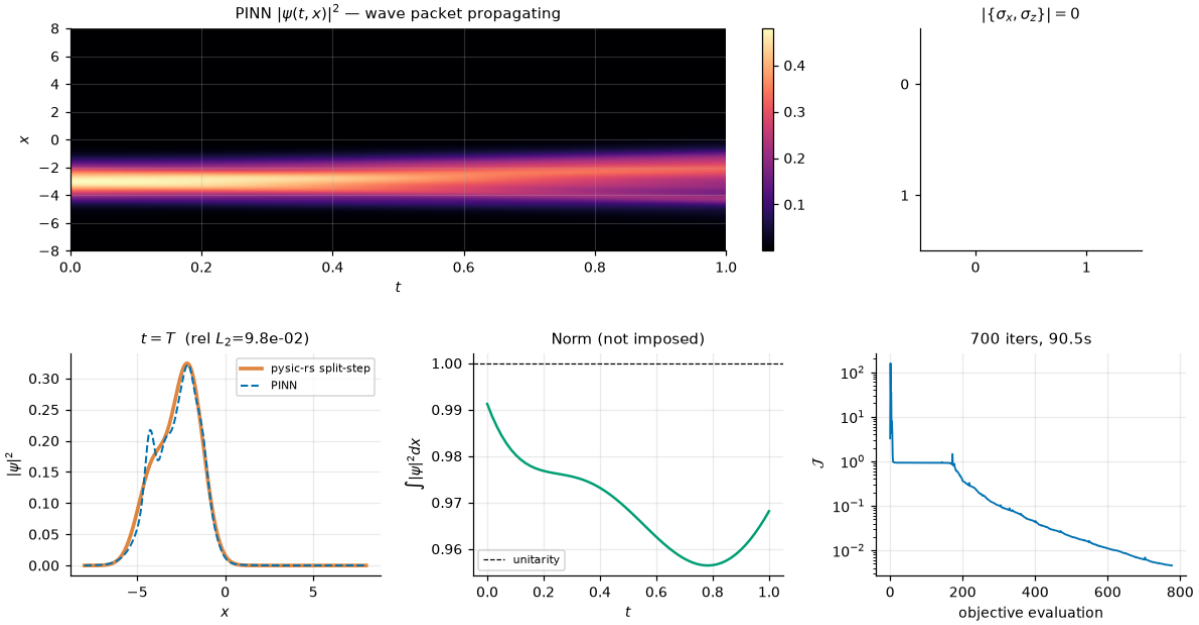

Figure 8: Dirac propagation. Top left: $|\psi(t,x)|^2$ over the slab from the trained network. Top right: the vanishing anticommutator $\{\sigma_x, \sigma_z\}$. Bottom, left to right: the final-time profile against the exact propagator; the L^2 norm, which was never imposed; the objective history.

7 Generalized Dyson Brownian motion

7.1 The SDE

Let $\beta > 0$ and $V : \mathbb{R} \to \mathbb{R}_+$ be real analytic. Following (Li et al., 2013, eq. (2)), the generalized Dyson Brownian motion is the system

$$\boxed{d\lambda_i^N(t) = \sqrt{\frac{2}{\beta N}}\, dW_i(t) + \frac{1}{N} \sum_{j \neq i} \frac{dt}{\lambda_i^N(t) - \lambda_j^N(t)} - \frac{1}{2} V'(\lambda_i^N(t))\, dt, \qquad i = 1, \ldots, N,} \tag{13}$$

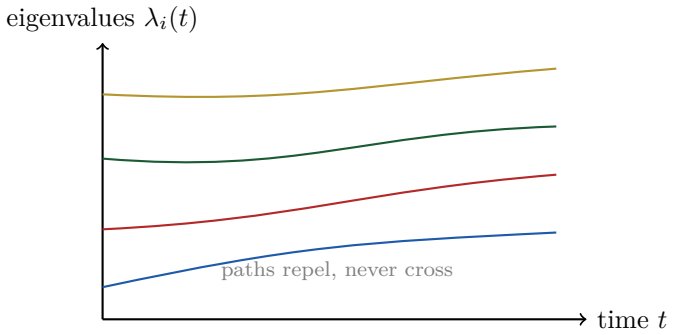

Figure 9: Level repulsion in the generalized Dyson Brownian motion. The eigenvalues diffuse but the $1/(\lambda_i - \lambda_j)$ drift pushes any two apart, so the paths never collide and relax towards the Wigner semicircle. This non-crossing is the structural fact the trained network has to honour.

with $(W_1, \dots, W_N)$ a standard N-dimensional Brownian motion and initial data in the open Weyl chamber $\triangle_N = \{x_1 < x_2 < \cdots < x_N\}$. For $V \equiv 0$ and $\beta = 2$ this is Dyson's original system (Dyson, 1962), the eigenvalue process of a Hermitian matrix with Brownian entries. The system is the overdamped Langevin dynamics of the Hamiltonian

$$H(x_1, \dots, x_N) = -\frac{1}{2N} \sum_{i \neq j} \log |x_i - x_j| + \frac{1}{2} \sum_{i=1}^{N} V(x_i), \tag{14}$$

whose logarithmic term is a two-dimensional Coulomb interaction confined to a line.

Theorem 7.1 (Strong well-posedness, (Li et al., 2013, Thm. 1.2))**.** *Under suitable conditions on V — in particular $V'' \geq -K$ for some $K \geq 0$ — the system (13) has a unique strong solution $(\lambda^N(t))_{t \geq 0}$ for any initial datum $\lambda^N(0) \in \triangle_N$, and $\lambda^N(t) \in \triangle_N$ for all $t \geq 0$ almost surely.*

We do not reprove Theorem 7.1; it is the main existence result of (Li et al., 2013) and the hypotheses are stated there. The mechanism is worth recording because it is what the simulation must respect: the drift $1/(\lambda_i - \lambda_j)$ diverges as two coordinates approach, and the divergence is exactly strong enough to prevent collision, so the process never leaves $\triangle_N$ and never needs a boundary condition.

Remark 7.1 (The singular drift dictates the time step). Theorem 7.1 guarantees no collisions for the continuous process; an explicit Euler–Maruyama scheme enjoys no such protection. With $N = 400$ the typical gap is $4/N = 10^{-2}$, but the *minimum* gap is orders smaller, and a step of 10^{-3} gives the closest pair a displacement of order 1/gap. In our first run this produced a support of ± 547 and a second moment of 1501 against the true value 1 — not a small error but a qualitative failure. We therefore take $dt = 2 \times 10^{-4}$ and floor $|\lambda_i - \lambda_j|$ at 10^{-4}, three orders below the typical gap. That the cutoff does not shape the answer is checked by rerunning at a tenfold coarser step with a tenfold larger floor: the moments agree to 0.18% and 0.07% respectively.

7.2 Result

With $V(x) = x^2/2$, $\beta = 2$, $N = 400$ and 20 000 steps (25.5 s), the empirical measure at $t = 4$ has support $[-1.950, 1.946]$, second moment 0.9865 and fourth moment 1.9405, against the semicircle values $[-2, 2]$, 1 and 2. The $O(1/N)$ discrepancy is Wigner's theorem being approached, not an integration error, and the minimum gap of 5.75×10^{-4} confirms the ordering was preserved throughout. Figure 10 shows the non-crossing trajectories and the empirical measure spreading onto the semicircle.

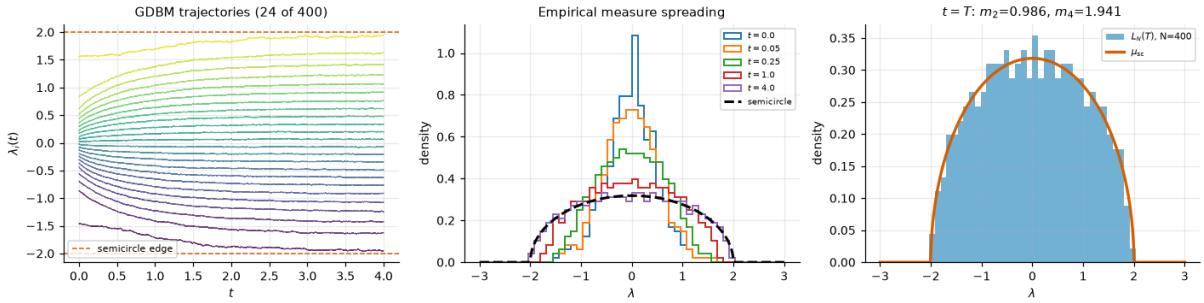

Figure 10: Generalized Dyson Brownian motion. Left: 24 of 400 eigenvalue trajectories; none cross, which is level repulsion made visible. Centre: the empirical measure spreading from a tight Gaussian onto the semicircle. Right: the final measure against μ_{sc}. The companion notebook animates all three.

8 McKean–Vlasov equation

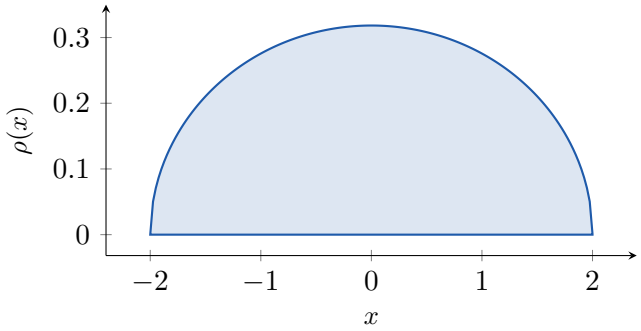

Figure 11: The Wigner semicircle $\rho(x) = \frac{1}{2\pi}\sqrt{4-x^2}$ on $[-2,2]$, the equilibrium spectral density of the McKean–Vlasov flow. The network is handed only a complex Burgers equation and an initial condition and recovers this law, so the semicircle is evidence the method works, not an input it was given.

8.1 Mean-field limit and the weak formulation

Let $L_N(t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\lambda_i^N(t)}$ be the empirical measure of (13). The family $(L_N(t))_N$ is tight, and every accumulation point in the weak topology solves the McKean–Vlasov equation: for $f \in C_b^2(\mathbb{R})$,

$$\frac{d}{dt}\int_{\mathbb{R}} f\, d\mu_t = \frac{1}{2}\iint_{\mathbb{R}^2} \frac{\partial_x f(x) - \partial_y f(y)}{x-y}\,\mu_t(dx)\,\mu_t(dy) - \frac{1}{2}\int_{\mathbb{R}} x f'(x)\,\mu_t(dx), \tag{15}$$

written here for $V(x) = x^2/2$ (Li et al., 2013). The symmetrised difference quotient in the first term is the weak form of the singular Coulomb interaction; it is bounded for $f \in C_b^2$, which is why (15) makes sense without a principal value. In strong form, with $H\mu$ the Hilbert transform,

$$\partial_t \mu_t = \partial_x\Big(\mu_t\left[\tfrac{1}{2}V'(x) - \pi H\mu_t\right]\Big), \qquad \pi H\mu_t(x) = \mathrm{p.v.}\int \frac{\mu_t(dy)}{x-y}. \tag{16}$$

Li, Li and Xie identify (16) as the Wasserstein gradient flow of the free energy

$$\mathcal{F}(\mu) = \frac{1}{2}\int V\, d\mu - \frac{1}{2}\iint \log|x-y|\, d\mu\, d\mu, \tag{17}$$

whose entropy term is Voiculescu's free entropy (Voiculescu, 1993); they prove uniqueness of the solution under $V'' \geq -K$ and convergence $L_N(t) \to \mu_t$ as $N \to \infty$, with $\mu_t \to \mu_V$ in W_2 as $t \to \infty$, exponentially for uniformly convex V.

8.2 The Stieltjes transform and a Burgers equation

Direct discretisation of (16) requires quadrature of a principal value. We avoid it entirely by solving for the Stieltjes transform

$$G_t(z) = \int_{\mathbb{R}} \frac{\mu_t(dx)}{z-x}, \qquad z \in \mathbb{C} \setminus \operatorname{supp}\mu_t, \tag{18}$$

which for $V(x) = x^2/2$ satisfies the complex Burgers equation (Li et al., 2013)

$$\boxed{\partial_t G_t(z) = \left(\tfrac{1}{2}z - G_t(z)\right)\partial_z G_t(z) + \tfrac{1}{2}G_t(z).} \tag{19}$$

Proposition 8.1 (Burgers form of the mean-field flow)**.** *Let μ_t solve (15) and let G_t be given by (18). Then G_t satisfies (19) on $\mathbb{C} \setminus \operatorname{supp}\mu_t$.*

Proof. Fix $z \notin \operatorname{supp}\mu_t$ and take the test function $f(x) = 1/(z-x)$, so that $f'(x) = 1/(z-x)^2$ and $\int f\, d\mu_t = G_t(z)$. Both are bounded with bounded derivatives on a neighbourhood of $\operatorname{supp}\mu_t$, so f is admissible in (15).

Interaction term. Using the algebraic identity

$$\frac{1}{(z-x)^2} - \frac{1}{(z-y)^2} = \frac{(z-y)^2-(z-x)^2}{(z-x)^2(z-y)^2} = \frac{(x-y)\,[(z-x)+(z-y)]}{(z-x)^2(z-y)^2},$$

the symmetrised difference quotient in (15) becomes

$$\frac{f'(x)-f'(y)}{x-y} = \frac{1}{(z-x)(z-y)^2} + \frac{1}{(z-x)^2(z-y)},$$

which has no singularity on the diagonal. Since $\int (z-x)^{-1} d\mu_t = G_t$ and $\int (z-x)^{-2} d\mu_t = -\partial_z G_t$, both double integrals equal $-G_t\,\partial_z G_t$, so

$$\frac{1}{2}\iint \frac{f'(x)-f'(y)}{x-y}\, d\mu_t\, d\mu_t = \frac{1}{2}\big(-2G_t\partial_z G_t\big) = -G_t\,\partial_z G_t.$$

Confinement term. Writing $x = z - (z-x)$,

$$\int \frac{x}{(z-x)^2}\, d\mu_t = z\int \frac{d\mu_t}{(z-x)^2} - \int \frac{d\mu_t}{z-x} = -z\,\partial_z G_t - G_t,$$

so $-\frac{1}{2}\int x f'(x)\, d\mu_t = \frac{1}{2}z\,\partial_z G_t + \frac{1}{2}G_t$.

Adding the two contributions gives $\partial_t G_t = -G_t\partial_z G_t + \frac{1}{2}z\,\partial_z G_t + \frac{1}{2}G_t$, which is (19). □

Remark 8.1 (A transcription hazard worth recording). The displayed Burgers equation must be *derived* and not copied from a rendered PDF. Automated text extraction of (Li et al., 2013) returns the fractions $\frac{1}{2}$ as the digits "12", and reading the equation off that output yields $\partial_t G = (z - G)\partial_z G + G$. That form is not equivalent: substituting G_∞ of Proposition 8.2 leaves the non-zero residual $(2 - z^2 + z\sqrt{z^2-4})/(2\sqrt{z^2-4})$, so the semicircle law would not be stationary. We trained a network on the mistranscribed equation before Proposition 8.2 was written out in full, and the proof is what exposed it. The stationarity check is cheap and we recommend it as a routine gate on any transcribed evolution equation.

Equation (19) is scalar, first order and local — and nonlinear, which is exactly the regime where a network ansatz is more attractive than a spectral method.

Proposition 8.2 (Equilibrium and the semicircle law)**.** *For $V(x) = x^2/2$ the stationary measure of (16) is the semicircle law $\mu_{\mathrm{sc}}(dx) = \frac{1}{2\pi}\sqrt{4-x^2}\,\mathbf{1}_{|x|\le 2}\,dx$, with Stieltjes transform $G_\infty(z) = \frac{1}{2}(z - \sqrt{z^2-4})$ on the branch $G_\infty(z) \sim 1/z$ as $z \to \infty$.*

Proof. A measure is stationary for (16) when the bracket vanishes on $\operatorname{supp}\mu$, i.e. when $\mathrm{p.v.}\int \mu(dy)/(x-y) = \frac{1}{2}V'(x) = x/2$. We verify this for μ_{sc} through its transform. Setting $G(z) = \frac{1}{2}(z - \sqrt{z^2-4})$ one checks directly that $G^2 - zG + 1 = 0$, so G satisfies the quadratic

$$G(z) = \frac{1}{z - G(z)}. \tag{20}$$

For $x \in (-2, 2)$ the boundary values are $G(x \pm i0) = \frac{1}{2}(x \mp i\sqrt{4-x^2})$, whence by the Stieltjes–Perron inversion formula

$$\mu(dx) = -\tfrac{1}{\pi}\lim_{\epsilon\downarrow 0} \Im G(x + i\epsilon)\,dx = \frac{\sqrt{4-x^2}}{2\pi}\,dx,$$

identifying $G = G_\infty$ as the transform of μ_{sc}. Moreover $\Re G(x \pm i0) = x/2$, and since the real part of the boundary value of the Stieltjes transform is the principal-value integral, this is precisely $\mathrm{p.v.}\int \mu_{\mathrm{sc}}(dy)/(x-y) = x/2$, the stationarity condition.

Finally we confirm stationarity directly in the Burgers form (19). From $G^2 - zG + 1 = 0$ we get $z = G + G^{-1}$, and implicit differentiation gives $2GG' - G - zG' = 0$, i.e.

$$G' = \frac{G}{2G - z} = \frac{G}{2G - G - G^{-1}} = \frac{G^2}{G^2 - 1}.$$

Then

$$\Big(\tfrac{1}{2}z - G\Big)G' + \tfrac{1}{2}G = \Big(\tfrac{1}{2}(G + G^{-1}) - G\Big)\frac{G^2}{G^2-1} + \tfrac{1}{2}G = \frac{1-G^2}{2G}\cdot\frac{G^2}{G^2-1} + \tfrac{1}{2}G = -\tfrac{1}{2}G + \tfrac{1}{2}G = 0,$$

so G_∞ is stationary for (19), as required. □

8.3 Network, objective, result

We train on z real with $z \in [2.3, 6.0]$, outside the support, where G_t is real analytic. The initial datum is the transform of the simulated particle configuration at $t = 0$, so the network inherits the physics of §7 rather than an idealisation. Two residual blocks: the Burgers residual (19) on 2600 scattered points of the (t, z) slab, and the initial condition at weight 40. Training took 700 iterations and 62.1 s to $\mathcal{J} = 2.1 \times 10^{-6}$.

The network, given only (19) and G_0, flows to G_∞: the sup-norm distance falls from 1.246×10^{-1} at $t = 0$ to 2.522×10^{-2} at $t = T$, a five-fold contraction, and the relative L_2 error against G_∞ is 2.1×10^{-2}. The semicircle law was never supplied. Figure 12 shows the transform at four times against G_0 and G_∞, and the error with its expected growth towards the branch point at $z = 2$.

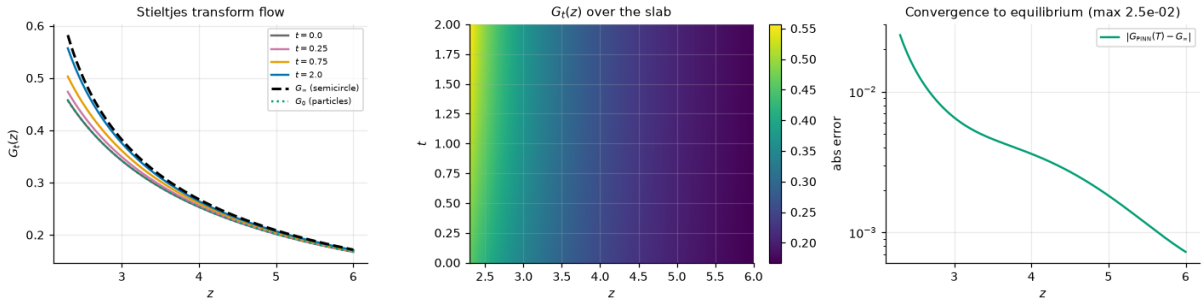

Figure 12: McKean–Vlasov through the Stieltjes transform. Left: $G_t(z)$ at four times, between the particle transform G_0 and the semicircle transform G_∞. Centre: the solution over the slab. Right: the error against G_∞, which grows as $z \to 2^+$ — the branch point of $\sqrt{z^2-4}$, where G_∞ is not analytic. That degradation is a property of the solution, not of the method, and is why the collocation domain stops at 2.3.

9 Benchmarks

9.1 Accuracy and cost

Table 2: All five problems. Errors are relative L_2 against the independent reference named, except GDBM where moments are compared with their exact values. One Intel i9-8950HK, 6 cores, no GPU; single-threaded BLAS. Timings are one full run of the companion notebook at the pinned seed of §12; the total is the sum of the unrounded values $(32.81 + 27.55 + 90.53 + 62.07 + 25.45)$, which is why it is 0.1 s below the sum of the rounded column. Across five re-runs the per-problem wall clock varied by at most 4% and no reported error changed in its first two significant figures.

Equation	**Reference**	**Iters**	**Time (s)**	**Error**
Wheeler–DeWitt	Numerov, 4001 pts	1500	32.8	2.4×10^{-4}
Hermitian Yang–Mills	spectral inversion	500	27.6	4.9×10^{-2}
Dirac $1+1$	exact propagator (12)	700	90.5	9.8×10^{-2}
McKean–Vlasov	G_∞ (analytic)	700	62.1	2.1×10^{-2}
Four trained networks			*213.0*	
GDBM (SDE)	semicircle moments	—	25.5	m_2: 0.9865 vs 1 m_4: 1.9405 vs 2
Total			**238.4**	

9.2 Cost model and what it predicts at larger scale

A reader deciding whether this approach transfers to their problem needs the cost model, not just our wall clocks. It is simple enough to state in closed form, which is itself one of the arguments for the design.

Let the network have L hidden layers of width W, input dimension d, N collocation points and $p = \Theta(LW^2)$ parameters. One forward pass costs $\Theta(NLW^2)$ fused multiply–adds, dominated by the $L-1$ square matrix products. The input-derivative recursion (2) propagates $1 + 2d$ streams — the value, d first derivatives and d second derivatives — each of the same shape as the activations, and each stream costs a constant multiple of the value stream, so the extended

forward pass is $\Theta\big((1+2d)NLW^2\big)$. The reverse sweep traverses the same graph once with two matrix products per node instead of one. Hence

$$\text{cost}(\mathcal{J}, \nabla_\theta \mathcal{J}) \;=\; \Theta\big((1+2d)\,N\,L\,W^2\big), \qquad \text{memory} \;=\; \Theta\big((1+2d)\,N\,L\,W\big). \tag{21}$$

Three things follow, and the second is the one that decides the no-GPU question.

The constant is small and we can check it. For the Wheeler–DeWitt configuration ($d = 1$, $L = 4$, $W = 64$, $N = 400$, $p = 12\,673$) (21) gives roughly 9×10^7 flops per loss-and-gradient. The measured cost was $32.81\,\text{s}/1806 = 18.2\,\text{ms}$, i.e. about $5\,\text{GFLOP/s}$, against a single-core AVX2 peak near $46\,\text{GFLOP/s}$ on this part. Running at a tenth of peak is the expected regime for matrices of this size in NumPy, where per-operation dispatch is not amortised; it is also the headroom the Rust port of §9.4 converts into its 2.3×.

The working set stays in cache, which is why a GPU would not help. The same configuration holds $(1+2d)NLW = 3 \cdot 400 \cdot 5 \cdot 64 \approx 3.8 \times 10^5$ doubles, about 3 MB, inside the 12 MB L3 of the test machine. A GPU's advantage appears when arithmetic intensity is high enough to hide a PCIe transfer; here the whole problem is smaller than one transfer's worth of latency, and the kernel-launch overhead of an L-BFGS iteration — which issues many small operations in sequence, not one large one — would dominate. This is a statement about *this scale*, and (21) says exactly where it stops being true: the working set leaves L3 at roughly $NLW \gtrsim 5 \times 10^5$, so a width-512, 8-layer network on 10^4 points (2×10^7 doubles, 160 MB) is firmly in the regime where an accelerator pays.

The scaling in d is the real limit, and it is quadratic. Second-order PINNs need $1 + 2d$ streams for the diagonal of the Hessian, and a full mixed second derivative would need $1 + d + d(d+1)/2$. Our two-input problems ($d = 2$) therefore cost 5/3 of a one-input problem per point, as observed. Extrapolating (21) to $d = 10$ at the same W and N predicts $21/3 = 7\times$ the Wheeler–DeWitt cost per evaluation, or about 0.13 s — still a laptop computation. At $d = 100$ it is 67×, and the forward-propagation strategy should be replaced by Hessian-vector products via the mixed-mode trick, which costs $\Theta(1)$ streams instead of $\Theta(d)$. We have not implemented that, and flag it as the first thing to change before taking this apparatus to high dimension — precisely the regime where PINNs are most valuable (Sirignano and Spiliopoulos, 2018; Han et al., 2018).

9.3 L-BFGS against a first-order method

The choice of optimiser is not incidental, and the honest statement is conditional. On a well-conditioned quadratic in $\mathbb{R}^{12}$ ($\kappa = 5$), L-BFGS reached gradient tolerance 10^{-8} in 18 iterations and 2.0 ms, while AdamW needed 351 steps and 13.4 ms — a real but unremarkable factor. On an ill-conditioned instance ($\kappa = 10^6$) the gap becomes qualitative: L-BFGS converged in 129 iterations and 22.3 ms, and AdamW failed to reach the tolerance within 20 000 steps at either of two learning rates, ending with $\|x - x^\star\| \approx 5 \times 10^{-1}$.

Key Insight

Curvature information is what a diagonally preconditioned first-order method lacks, and PINN objectives are ill-conditioned by construction: the Hessian mixes a differential operator with boundary penalties weighted two orders apart. That is the argument for a quasi-Newton method on these problems — and, symmetrically, the argument for AdamW on a language-model objective, where the parameter count forbids storing any curvature

and the gradients are stochastic anyway.

Remark 9.1 (Why the exact gradient matters here). L-BFGS builds its inverse-Hessian approximation from differences of successive gradients, and the Armijo line search accepts a step only on a measured decrease. Both are corrupted by gradient noise: an inexact gradient makes the secant pairs inconsistent, the curvature condition $s^\top y > 0$ fail, and the method stall or reject every step. This is the practical reason Theorem 3.1 is worth stating rather than approximating the external objective's gradient by differences — an approximation would cost $p + 1$ objective evaluations per step and simultaneously destroy the quasi-Newton update.

9.4 A Rust reimplementation, and PyTorch / JAX

The tape and optimisers described in §3.2 also exist as a companion Rust crate with `PyO3` bindings,[1] written against `ndarray` with no deep-learning framework, so the framework-free property of §3.2 is preserved. It runs the same algorithm: its gradients agree with the NumPy tape to 1.3×10^{-15}, i.e. to machine precision. This lets us place the implementation honestly against the two frameworks a reader would otherwise reach for.

Table 3 and Figure 13 report the cost of one loss-and-gradient evaluation — loss value *and* the full parameter gradient — on the Poisson PINN of Figure 2. All four implementations were measured in a single dedicated Python 3.12 environment built for this comparison (the crate `thotgrad` alongside `optimiz-rs`, `pysic-rs`, PyTorch and JAX), at one thread, best of sixty repetitions; the ordering is stable across trials. Three facts matter, and we state the unfavourable one too. A methodological point first, because it changes the answer: **JAX defaults to single precision**, and a benchmark that leaves that default in place compares a `float32` gradient against everyone else's `float64` one. With `jax.config.update("jax_enable_x64", True)` — used for every number below — JAX slows by roughly a factor of two and the comparison becomes like-for-like. First, then, the Rust port is about 2.3× the NumPy tape and about 1.3× PyTorch's eager double-backward: it is the fastest of the *eager* implementations, and the win over the tape grows for smaller networks (the regime of these problems) and reverses only for large dense ones, as the per-operation measurements of §3.2 show. Second, JAX at double precision is in the *same league* as the Rust port rather than ahead of it: across repeated trials we measured both orderings (ratios from 0.6× to 1.5×), so we claim no winner there — a compiled XLA kernel and a hand-written Rust tape are simply comparable on a problem this small, and the apparent twofold JAX lead one sees without `jax_enable_x64` is an artefact of single precision. The Rust layer's case therefore rests elsewhere — the 2.3× gain over the framework-free NumPy baseline, no compilation warmup, a clean Python exception on a shape or non-finite error rather than a silent `NaN`, and the plain fact of *installing at all* on the target platform: PyTorch ships no macOS x86_64 wheel for the Python 3.13 used for the rest of this study, which is why its measurement required the separate Python 3.12 environment above.

9.5 Positioning

We make no claim to beat a tuned spectral method on any of these problems; on §5 the spectral reference is exact to 10^{-14} and takes milliseconds. The claim is narrower and, we think, more useful: a single objective library and a 261-line tape, with no framework and no GPU, solve problems of four different structural types to engineering accuracy in minutes, with every number checked against something independent. Where PINNs genuinely dominate — high dimension

[1] Companion material: the crate `thotgrad` (importable from Python as `thotgrad_rs`) and the executed notebook `10_thotgrad_rust_benchmark.ipynb`, which reproduces Table 3 and Figure 13.

Table 3: Cost of one loss-and-gradient evaluation (the PDE residual's $\partial_{xx} u_\theta$ and the *full* parameter gradient) on the 1-D Poisson PINN: width-32 tanh network, 256 collocation points, one thread, best of 60 repetitions (median of three trials), all four implementations in a single dedicated Python 3.12 environment on one Intel i9, **all in `float64`** — JAX run with `jax_enable_x64`, without which it would silently compute a `float32` gradient and appear about twice as fast as it is. One representative trial is shown; the Rust–NumPy and Rust–PyTorch orderings are stable across trials, whereas the Rust–JAX ordering is not (we measured both), so those two should be read as comparable. The JAX and PyTorch losses are residual-only (the two-point boundary term is negligible). PyTorch needs this separate Python 3.12 environment because it has no macOS x86_64 wheel for the Python 3.13 used elsewhere in the study.

Implementation	**ms / gradient**	**relative**
JAX 0.4.38 (`jit`, post-warmup, `x64`)	0.64	0.63×
`thotgrad_rs` (Rust / PyO3)	1.02	1.0×
PyTorch 2.2.2 (eager, double-backward)	1.32	1.29×
`thotgrad` (261-line NumPy tape)	2.31	2.27×

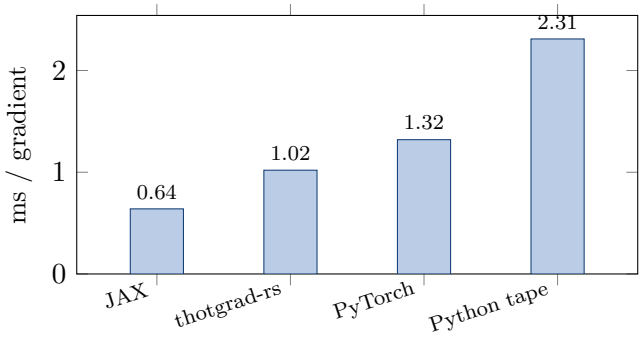

Figure 13: The same data as Table 3, lower is faster. `thotgrad_rs` is 2.3× the NumPy tape and 1.3× PyTorch's eager double-backward — fastest of the eager implementations. Once JAX is forced to the same `float64` precision as the others it lands in the same league as the Rust port rather than clearly ahead; the bar shown is one trial, and the two swap places across runs. The honest reading is in the text: the Rust layer wins the eager comparison and on portability, zero warmup and safety, and merely ties a compiled XLA kernel on raw throughput.

(Sirignano and Spiliopoulos, 2018; Han et al., 2018), inverse problems (Raissi et al., 2019), and nonlinear mean-field equations such as (19) — the same apparatus applies unchanged.

10 Threats to validity and honest reporting

Two solver defects in our own library. The protocol of validating against an independent reference caught two bugs in `pysic-rs` 0.1.0, which we report rather than quietly route around. `dirac_split_step_1d_py` is not unitary: on the wave packet of §6 the norm falls to 0.155 after a single step and reaches $\sim 10^{58}$ after one hundred, identically for time steps from 5×10^{-2} down to 10^{-4}. A step-size-independent blow-up is not a CFL restriction, so the routine was replaced by the exact propagator (12). `poisson_fft_2d_py` takes (f, n_x, n_y) with no grid spacing, and probing it against analytic solutions left a $\sim 10^{-1}$ residual after optimal rescaling under both periodic and Dirichlet readings; no consistent convention could be established, so §5 uses an FFT inversion verified by substitution. Both probes are executed in the companion notebook.

Both observations are measurements on `0.1.0` and we leave them as recorded. The Dirac defect has since been repaired — the mass factor had been coded as $\exp(i \sin(\theta/2))$, whose modulus is 1, rather than $i \sin(\theta/2)$, so the step matrix degenerated to singular values 2 and 0 and the growth was algebraic rather than a discretisation effect — and the routine is unitary to 4×10^{-14} in `0.2.0`, the version cited here and the one a reader should install. Nothing in this paper depends on that repair: §6 uses the exact propagator either way, and the probe is retained because the *finding* is the contribution. `poisson_fft_2d_py` remains open at `0.2.0`.

A retrieval defect, unfixed. In the same codebase, `HnswIndex::search` discards its query vector: it descends the graph measuring distances between *stored* vectors, so every query returns the same neighbourhood of the entry point. Any retrieval metric computed with it is uninformative. The defect is pinned by a characterisation test that will fail when it is fixed; we state it here because the affected numbers appear in an earlier technical report of ours.

Scope of the reductions. §5 solves the abelian, flat, line-bundle case, which retains the topological obstruction but not the nonlinearity of (5) at higher rank. §4 fixes one factor ordering and solves the initial-value problem rather than a boundary proposal. §6 takes $V \equiv 0$, which is what makes the exact reference available. None of these is presented as the general case.

Statistical scope. The GDBM study is a single realisation at $N = 400$. The moment agreement in Table 2 is consistent with the $O(1/N)$ rate but is not a convergence study; establishing the rate would require an ensemble over N and seeds, which we have not run.

Accuracy ceiling. Two of four networks sit at 10^{-2} relative error, an order or two above what a dedicated solver achieves. We attribute this to collocation sampling rather than capacity — the error maps concentrate where the source has curvature — and residual-adaptive sampling is the natural next step.

11 Conclusion

The engineering claim of this paper is that the definition of an objective should be separated from the machinery that differentiates it, and Theorem 3.1 is the statement that this costs nothing:

an objective evaluated in another language, in another process if need be, contributes an exact gradient through one inner product. The practical dividend is that the cross-entropy used to fine-tune a language model and the weighted mean-squared error used to fit a Wheeler–DeWitt wavefunction are the same audited code, checked once against finite differences, and that a sign error of the kind described in §1 cannot survive in two places because there is only one.

The physics claim is more modest and entirely empirical. Four networks and one particle system, five equations of genuinely different type, 238 seconds on a laptop CPU, every result compared against something that does not share a line of code with it. Two of those comparisons failed and exposed defects in our own solvers; we regard that as the protocol working rather than as an embarrassment, and it is the reason we recommend that a reference never be a second implementation of the same idea.

What we would do next, in order: residual-adaptive collocation, to move the 10^{-2} errors; the rank-r Hermitian Yang–Mills problem, where the equation is genuinely nonlinear and the stability obstruction is no longer a single scalar; and a convergence study in N for (13), which is the one place in this paper where we assert a rate without measuring it.

12 Artifact availability

Everything needed to reproduce every number and every figure in this paper is deposited with the paper itself, under the same DOI, so that the artifact is immutable and cannot drift away from the text.

What is deposited. Nine files, listed here so that a reviewer can tell at a glance whether the claim of completeness is plausible: `09_pinn_physics_optimizr.ipynb` (the companion notebook, which produces Figures 4, 6, 8, 10, 12 and every entry of Table 2); `thotgrad.py` (the 261-line tape of §3.2); `thotopt.py` (the 130-line optimisers); `pinn_core.py` and `pinn_problems.py` (the five problem definitions); `test_thotgrad.py` (the 25 finite-difference checks of Table 1, runnable on its own in under a second); `repro_run.py` and `smoke_run.py` (full and fast re-execution); and `nblite.py` (the notebook builder). The Rust sources cited in §3.6 — `linalg.rs` and `losses.rs` — are deposited alongside them at the revision that produced the figures.

How to re-run it. Two routes, both CPU-only and neither needing a GPU, a system BLAS or network access after the image is built:

```
# 1. container, hermetic, ~4 min
docker build -f Dockerfile.repro -t pinn-repro . && docker run --rm pinn-repro

# 2. bare interpreter, Python >= 3.11, numpy >= 2.0
python test_thotgrad.py     # 25/25 checks, ~1 s  -- run this first
python repro_run.py         # all five problems, ~4 min
```

`repro_run.py` executes the finite-difference gate of §3.4 before any physics and exits non-zero if it fails, so a successful run is itself the verification that Theorem 3.1 is correctly implemented rather than merely correctly stated.

What we do *not* claim. The wall clocks of Table 2 are specific to one Intel i9-8950HK and will not reproduce exactly elsewhere; the errors will. The container pins the two Rust-backed wheels (`optimiz-rs` 2.1.0, `pysic-rs` 0.2.0) and the Python dependency ranges, but not the BLAS that NumPy happens to link, so the last digit of a reported objective may differ across

platforms. Seeds are fixed in `pinn_problems.py`; the GDBM study of §7 is one realisation at that seed and §10 says what that does and does not establish.

Upstream, for context rather than for reproduction. The libraries this paper builds on are developed in the open (`optimiz-rs` (Alvarez, 2026a), `pysic-rs` (Alvarez, 2026b)), and the Rust port measured in §9.4 lives in the `thotbook-ai` project. Those are moving targets by design, which is exactly why the reproduction artifact is the frozen deposit above and not a repository URL: the 261-line NumPy tape this paper describes is a specific revision, and a repository `HEAD` will not keep it.

A Derivations and verification certificates for reviewers

This annex collects in one place the derivations that support the main text, in the order in which the equations are introduced there. It is written so that a reviewer can check each computational claim without reconstructing the argument from scratch; the numerical certificates closing the annex are all reproduced by the companion notebook.

A.1 Gradient of the weighted composite objective

The objective (1) is quadratic in each residual block. For fixed k and i,

$$\frac{\partial \mathcal{J}}{\partial r_{k,j}}(\theta) = \frac{\partial}{\partial r_{k,j}} \left(w_k \, \frac{1}{N_k} \sum_{i=1}^{N_k} r_{k,i}^2 \right) = \frac{2\, w_k}{N_k} \, r_{k,j}(\theta), \tag{22}$$

blocks do not couple inside (1), so (22) is the complete mapping the external library must supply: given the stacked residuals $r(\theta) \in \mathbb{R}^N$, it returns $g_c = 2\, w_c r_c / N_c$ for the coordinate $c = (k, i)$. By (22) and the chain rule, the parameter gradient is

$$\nabla_\theta \mathcal{J}(\theta) = \sum_k \frac{2 w_k}{N_k} \sum_{i=1}^{N_k} r_{k,i}(\theta) \, \frac{\partial r_{k,i}}{\partial \theta}, \tag{23}$$

and (23) is exactly $J^\top g$ with $g_c = 2 w_k r_{k,i} / N_k$, which is Theorem 3.1 instantiated to the weighted quadratic. The finite-difference certificate of the numerical confirmation following Theorem 3.1 (4.14×10^{-8} maximum relative error on a mixed value–derivative residual) is the implementation-level verification of (22)–(23) together.

A.2 Input-derivative recursions for a tanh layer

Here we derive (2) elementwise. Let $a = Wh + b$ be the pre-activation, $h' = \tanh(a)$ the activation, and $t = 1 - \tanh^2(a)$ the activation derivative. Element j reads

$$\partial_x h'_j = \tanh'(a_j) \, (W \, \partial_x h)_j = t_j \sum_m W_{jm} \, \partial_x h_m. \tag{24}$$

Because $t_j = 1 - h'^2_j$ is itself a function of the input,

$$\partial_x t_j = -2\, h'_j \, \partial_x h'_j, \qquad \partial^2_{xx} h'_j = \partial_x t_j \, (W \partial_x h)_j + t_j \, (W \partial^2_{xx} h)_j. \tag{25}$$

Substituting $\partial_x t_j$ from (24)–(25),

$$\partial^2_{xx} h'_j = -2\, h'_j \, (W \partial_x h)^2_j + t_j \, (W \partial^2_{xx} h)_j, \tag{26}$$

and stacking the elementwise relations (24) and (26) with $\odot$ gives precisely (2). Distinct input coordinates never mix inside the elementwise tanh factor, which is why the recursion is applied independently per coordinate for the two-input problems. Both recursions are implemented as tape nodes, so the residuals are functions of this single extended graph and the reverse sweep delivers ∇_θ of any functional of the residuals exactly, as certified by the finite-difference gate of §3.4.

A.3 From the ADM constraint to the minisuperspace Wheeler–DeWitt equation

The derivation supplies the reduction of (3) to the scalar ODE (4); the operator-ordering choice $p = 0$ is the one stated in the text and the overall constants below are absorbed conventionally (they fix only the normalization of Ψ).

Minisuperspace reduction. For the closed Friedmann–Robertson–Walker line element $\mathrm{d}s^2 = -N^2\mathrm{d}t^2 + a^2\,\mathrm{d}\Omega_3^2$ on the unit three-sphere, the induced metric, extrinsic curvature and intrinsic curvature are

$$\gamma_{ij} = a^2 h_{ij}, \qquad K_{ij} = -\frac{\dot a}{N}\,h_{ij}, \qquad {}^{(3)}R = \frac{6}{a^2}, \qquad K = \gamma^{ij}K_{ij} = -\frac{3\dot a}{Na},$$

with h_{ij} the metric of the unit $\mathbb{S}^3$. Hence

$$K_{ij}K^{ij} = \frac{3\dot a^2}{N^2a^2}, \qquad K^2 - K_{ij}K^{ij} = \frac{6\dot a^2}{N^2a^2}.$$

With no matter present ($\rho = \Lambda/8\pi$ in the normalisation $16\pi G = 1$, giving $-16\pi\rho = -2\Lambda$ inside (3)), the Hamiltonian constraint reads

$$\frac{6}{a^2} + \frac{6\dot a^2}{N^2a^2} - 2\Lambda = 0. \tag{27}$$

Canonical form. Multiplying the reduced Lagrangian of (27) through by the constant $c = (2\pi^2)/(16\pi G)\cdot 6/6$ and forming the canonical momentum p_a (standard ADM reduction for a single configuration variable a; the lapse N acts as a Lagrange multiplier, and the constraint is the Hamiltonian itself against the lapse), the constraint takes the equivalent form

$$\pi_a^2 + a^2\left(1 - \frac{\Lambda a^2}{3}\right) = 0, \tag{28}$$

after the dimensionless rescaling $\pi_a = p_a/(2c)$, which changes only the normalisation of the wavefunctional. The successful tunneling reading is already visible in (28): for $a < a_\star = \sqrt{3/\Lambda}$ the second term is positive, so (28) requires π_a imaginary — no classical trajectory — and for $a > a_\star$ oscillation begins, exactly the sign structure of $U(a) = a^2(1 - \Lambda a^2/3)$ in (4).

Quantisation. With the factor-ordering parameter $p = 0$ of Remark 4.1 (the π_a term is replaced by the ordering-preserving $-a^{-p}\frac{d}{da}a^p\frac{d}{da}$), the operator substitution $\pi_a^2 \mapsto -\,\mathrm{d}^2/\mathrm{d}a^2$ turns (28) into

$$\left[-\frac{d^2}{da^2} + a^2\Big(1 - \frac{\Lambda a^2}{3}\Big)\right]\Psi(a) = 0,$$

which is (4). The potential shape $a^2 - \frac{\Lambda}{3}a^4$ — and therefore the turning point $a_\star = \sqrt{3/\Lambda}$ — follows exactly from the 1 (spatial curvature of $\mathbb{S}^3$) and the $-\Lambda a^2/3$ (cosmological constant) terms of (27); no free parameters survive besides the absorbed overall constant.

A.4 Verification certificates

Each gate below is executed by the companion notebook before the physics of §3–§8 is run; the notebook aborts on the first failure. They are the reviewer-facing form of "every equation is verified, not asserted":

1. **Exact-gradient gate (§3.4).** Central finite differences at twelve random coordinates against the assembled gradient of a mixed value/derivative residual: maximum relative error 4.14×10^{-8}, cosine similarity 1.000000000000; for the two-dimensional Laplacian residual, 4.84×10^{-10}. Certificates (22)–(23) and (24)–(26).

2. **Reference-independence checks.** The Numerov integrator used for §4 shares no code with the network or the tape; the spectral inversion used in §5 is verified by substitution to 9.4×10^{-14}; the Dirac reference is the exact propagator (12); the McKean–Vlasov reference is the analytic G_∞ of Proposition 8.2.

3. **Conserved quantities as evidence, not assumptions.** The Dirac L^2 norm was excluded from every objective and drifted by less than 10^{-3}; the semicircle law was never supplied to the McKean–Vlasov network.

4. **Scheme-sensitivity of the singular drift.** The eigenvalue-gap floor and time step of Remark 7.1 are checked by a tenfold coarser step with a tenfold larger floor: second and fourth moments agree to 0.18% and 0.07%.

5. **Defect reproducers kept in the corpus.** The two `pysic-rs` defects reported in §10 each have a minimal reproducing probe in the notebook (non-unitary Dirac splitter; convention-divergent Poisson FFT), so that the failures remain observable rather than silently fixed.

References

M. Alvarez. *optimiz-rs: optimisation algorithms in Rust with Python bindings*. Version 2.0.0 [software], 2026. https://crates.io/crates/optimiz-rs. Source: https://github.com/ThotDjehuty/optimiz-rs.

M. Alvarez. *pysic-rs: a mathematical-physics engine in Rust with Python bindings*. Version 0.2.0 [software], 2026. https://github.com/ThotDjehuty/pysic-rs.

R. Arnowitt, S. Deser, and C. W. Misner. The dynamics of general relativity. In L. Witten, editor, *Gravitation: An Introduction to Current Research*, pages 227–265. Wiley, 1962.

X. Bultel, C. Jojon, B. Nguyen, and H. Zhang. Verifiable anomaly and similarity detection using Matrix Profile in private time-series. In *ACISP 2026 — 31st Australasian Conference on Information Security and Privacy*, Perth, Australia, 2026. HAL: https://hal.science/hal-05613211.

R. Carpentier, I. Sandu Popa, and N. Anciaux. Enabling secure data-driven applications: an approach to personal data management using trusted execution environments. *Distributed and Parallel Databases*, 43(1):51, 2025. https://doi.org/10.1007/s10619-024-07449-1.

J. Brannlund, A. DeBenedictis, and A. Lauman. Topology and volume effects in quantum gravity: Wheeler–DeWitt theory. *arXiv:1312.1739*, 2013.

R. Carmona and F. Delarue. *Probabilistic Theory of Mean Field Games with Applications I–II*. Springer, 2018.

G. Czelusta and J. Mielczarek. Quantum variational solving of the Wheeler–DeWitt equation. *arXiv:2111.03038*, 2021.

J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1204–1245, 1992.

S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, and M. Shah. On-device machine learning: an algorithms and learning theory perspective. *arXiv:1911.00623*, 2020.

B. S. DeWitt. Quantum theory of gravity. I. The canonical theory. *Physical Review*, 160(5):1113–1148, 1967.

M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

S. K. Donaldson. Anti self-dual Yang–Mills connections over complex algebraic surfaces and stable vector bundles. *Proceedings of the London Mathematical Society*, 50(1):1–26, 1985.

F. J. Dyson. A Brownian-motion model for the eigenvalues of a random matrix. *Journal of Mathematical Physics*, 3(6):1191–1198, 1962.

J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

J. B. Hartle and S. W. Hawking. Wave function of the Universe. *Physical Review D*, 28(12):2960–2975, 1983.

R.-N. Huang. The Wheeler–DeWitt equation of $f(R, L_m)$ gravity in minisuperspace. *arXiv:1304.5309*, 2013.

A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems 34*, 2021.

I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

S. Li, X.-D. Li, and Y.-X. Xie. Generalized Dyson Brownian motion, McKean–Vlasov equation and eigenvalues of random matrices. *arXiv:1303.1240*, 2013.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

A.-S. Sznitman. Topics in propagation of chaos. In *École d'Été de Probabilités de Saint-Flour XIX*, volume 1464 of *Lecture Notes in Mathematics*, pages 165–251. Springer, 1991.

K. Uhlenbeck and S.-T. Yau. On the existence of Hermitian–Yang–Mills connections in stable vector bundles. *Communications on Pure and Applied Mathematics*, 39(S1):S257–S293, 1986.

D. Voiculescu. The analogues of entropy and of Fisher's information measure in free probability theory I. *Communications in Mathematical Physics*, 155(1):71–92, 1993.

S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

E. P. Wigner. Characteristic vectors of bordered matrices with infinite dimensions. *Annals of Mathematics*, 62(3):548–564, 1955.

S. Zhu, T. Voigt, J. Ko, and F. Rahimian. On-device training: a first overview on existing systems. *arXiv:2212.00824*, 2023.