Baller: fail-closed ball arithmetic for scientific computing

Matthew D. Schwartz
Department of Physics, Harvard University

October 1, 2026

This work is preliminary. Some content remains AI-generated and is still being validated.

Abstract

Scientific computing runs on floating-point arithmetic that fails silently: a wrong answer arrives styled exactly like a right one, sixteen digits, no grade attached. The failure mode is old: least-squares programs returning coefficients with hardly a correct digit were reported in 1967, and the National Institute of Standards and Technology maintains reference problems with vetted answers precisely because widely used statistical packages have disagreed with them, sometimes by every digit they printed. We present `baller`, a harness built on arbitrary-precision ball arithmetic in which every computation has exactly two outcomes: an enclosure, an interval proven to contain the true value, or a typed refusal that names the test that failed, the coordinate block or border it failed on, and the margin by which it failed. Nothing averages and nothing warns in passing; when the arithmetic cannot prove the claim, the harness halts and hands back a machine-readable receipt. The semantics is in production: handed the reported optimum of a hierarchical model with 5,592 free coordinates, the harness refused: the fit had converged at its own gradient tolerance, but in the curvature metric the residual was four orders of magnitude too loose to certify. Two Newton steps later it delivered the full result: a strict local minimum, existing and unique in an explicit box, verified at 192-bit precision in about 90 seconds. The hardening is part of the record: a sixteen-leg mutation-tested battery, three adversarial break rounds whose six blocking findings are now regression tests, and a catalogue of measured traps the harness now refuses. Two public exhibits, the NIST reference problems and a head-to-head trap suite against existing interval libraries, are registered, with results to follow. Wherever a number must be right rather than plausible, refusal is now a practical default.

Contents

1 Introduction **2**

2 Two outcomes: enclosures and refusals **3**

3 The harness **5**

4 Hardening as content **8**

5 Production **10**

6 The exhibit program **12**

7 What exists, and what was missing **13**

8 Outlook 14

1 Introduction

Nearly every quantitative conclusion in science now passes through numerical software, and numerical software has one famous vice: it does not know when it is wrong. A floating-point computation that has lost all of its accuracy returns a number in the same format, at the same speed, with the same silence as one that has lost none. The people consuming the number, who are usually not numerical analysts, have no way to tell the two apart, so the wrong number propagates into fits, tables, and published conclusions with its confidence intact. This paper describes an instrument built so that this cannot happen: a computing harness whose every answer is either proven or refused.

1.1 Silent failure has a paper trail

The failure mode is not hypothetical, and it was documented before most working scientists were born. In 1967 Longley computed, by hand and by desk calculator, the correct least-squares coefficients for a 16-observation economic regression, then submitted the same problem to the electronic computer programs then in general use; most returned coefficients with hardly a correct digit among them [1]. Three decades later the National Institute of Standards and Technology (NIST) institutionalized the lesson as the Statistical Reference Datasets, including linear-regression, analysis-of-variance, univariate, and nonlinear-regression problems, each published together with reference values computed in high-precision arithmetic, so that anyone can check what their package actually returns [2]. McCullough ran exactly that audit against the major statistical packages of the late 1990s and found failures across the board, including nonlinear problems on which packages returned solutions with zero accurate digits, without any indication that anything had gone wrong [3, 4]. The companion audit of econometric software reached the same verdict [5]. The audits changed some packages and created a small literature of follow-ups, but they did not change the architecture: thirty years on, production statistics and production science still run on 53-bit floats, convergence flags that measure the optimizer's patience rather than the answer's correctness, and no grade on any printed digit.

The remedy has existed, in principle, for even longer. Interval arithmetic, in which every quantity carries bounds and every operation propagates them outward, dates to Moore in 1966 [6]; Krawczyk showed in 1969 how an interval Newton step can prove that a true zero of a nonlinear system exists, and is unique, inside an explicit box [7]; the mature theory is collected in Neumaier's book and Rump's Acta Numerica survey [8–10]. The substrates are public and excellent: Arb provides arbitrary-precision ball arithmetic with rigorously bounded special functions [11, 12], INTLAB has served MATLAB users for twenty-seven years [13], and interval libraries now exist for Python, R, and Julia [14–16], with an IEEE standard governing the semantics [17]. Yet silent failure persists, because the substrates compute enclosures and then stop. When an enclosure comes back too wide to mean anything, or a hypothesis of the theorem cannot be checked, every one of these systems hands the object back to the caller, and the caller, trained by fifty years of floating point, presses on. The missing layer is policy, not arithmetic: the decision about what happens when rigor runs out.

1.2 This paper

`baller` is that policy layer, written down as software: an arbitrary-precision ball-arithmetic harness, built on Arb through python-flint, in which a computation has exactly two possible outcomes. Either it produces an enclosure, an interval proven to contain the true value, or it raises a typed refusal that names the test that failed, the coordinate block or border on which it failed, and the measured margin by which it failed, together with the precision and inputs in force at the time. The refusal is a first-class result with a machine-readable receipt, designed to be filed, compared, and acted on. There is no third outcome, no warning that scrolls past, and no path on which a wide or non-finite ball flows onward as if it were a number.

The package is small and concrete: five modules, just under five thousand lines across nineteen files, combining engines extracted from production computing pipelines with components written for the harness, all behind one integrity layer (Section 3). Its reach is set by one structural instrument: a block-arrow Krawczyk step that proves existence, uniqueness, and positive definiteness for stationary points of hierarchical objectives with thousands of coordinates in seconds (Section 3.4). Its credibility rests on three records: its hardening record, a sixteen-leg battery in which every check is itself tested by mutation, three adversarial break rounds whose blocking findings became regression tests, and a catalogue of measured traps (Section 4); and by production use, one consumer to date, whose first serious request the harness refused, correctly (Section 5). Two public exhibits are registered with designs fixed in advance (Section 6), the surrounding software landscape is surveyed honestly (Section 7), and Section 8 closes with availability and reach. Sections 2 through 4 assume no physics and no prior exposure to interval methods; a reader who fits models for a living can read the paper as a user's tour of what fail-closed arithmetic would do to their own pipeline.

2 Two outcomes: enclosures and refusals

The harness rests on one representation and one contract. The representation is standard and old; the design lives in the contract. Both fit on a page, and every later section leans on both, so they are stated here in a form readable with no background.

2.1 Ball arithmetic in one page

A *ball* is a pair of numbers, a midpoint m and a radius r, asserting that the true value x of the quantity it represents satisfies

$$x \in [\, m - r,\ m + r \,]. \tag{1}$$

Arithmetic on balls propagates the radii in the worst case, with rounding always outward, so the assertion (1) is preserved by every operation: if the inputs contain the truth, the output provably contains the truth [6, 11]. A ball is therefore a theorem, and its radius is an unconditional bound on the accumulated error of everything that produced it, carried by the arithmetic itself rather than estimated after the fact. The number of *certified digits* of a ball is $\log_{10}(|m|/r)$: digits of the midpoint that no admissible value inside the ball can change. The precision of the midpoint arithmetic is a knob (the work in this paper runs at a few hundred bits, chosen per task); the radius is not a knob, and cannot be argued with.

In plain terms, ball arithmetic replaces "here are sixteen digits" with "here are the digits I can prove, and here is the proof budget I have left." The cost is a constant-factor slowdown; the return is that nothing is overstated: operations that lose accuracy show the loss immediately as

radius growth. A worked miniature makes the character of the thing clear. Squaring the ball 0 ± 0.1 must produce a ball containing zero, and a correct implementation does; but the general power function in the underlying library routes through $\exp(n \log x)$, and on a ball containing zero it returns not-a-number rather than a wrong interval. The substrate reports faithfully at every step. It does not decide what happens next, though, and in every mainstream interval library the answer is: the NaN, or the uselessly wide ball, is returned to the caller.

2.2 Typed refusals and their receipts

`baller`'s contract closes that gap. Every harness operation ends in one of two typed outcomes,

$$\mathsf{run}(f, x) \longrightarrow \begin{cases} \mathsf{Enclosure}(m, r) & \text{all proof obligations met,} \\ \mathsf{Refusal}(\mathsf{type}; \mathsf{receipt}) & \text{otherwise,} \end{cases} \tag{2}$$

and the refusal branch is engineered with the same care as the success branch. A refusal is *typed*: a non-finite result, a radius above the caller's stated cap, a containment test that could not be decided, a violated oracle contract, and a tampered component each raise a distinct exception class, so that policy can dispatch on what actually happened. Out-of-contract inputs are a separate, smaller class: a call the contract does not admit — a ball handed to a scalar comparison, a complex value where the bar is per-component — draws a typed *rejection* before any proof is attempted, so an input error is never mistaken for a proof outcome. And a refusal carries a *receipt*: the name of the failed test, the coordinate block or border it failed on, the measured value of the quantity that had to satisfy a bound (for example, the contraction factor that had to be provably below one and evaluated to 4.7×10^{10}), the arithmetic precision in force, and enough input identity to reproduce the attempt. A refusal with a receipt amounts to a measurement of where rigor ran out, and Section 5 shows one earning its keep.

An enclosure from the harness asserts a precise conditional statement, and the conditions belong in one place. Every enclosure is a theorem conditional on three things: the correctness of the underlying ball-arithmetic substrate (Arb and its Python bindings [11, 14]); the caller's oracle honoring its stated contract, namely that the function and derivative values it supplies are themselves rigorous enclosures over the whole input box; and the integrity of the running process within a stated boundary (Section 3.2). The second condition has a sharp consequence for the local-minimum instrument: the existence-uniqueness step is provably insensitive to certain oracle lies (a sign flip applied after the fact to a derivative block leaves the zero set unchanged), so oracle truth must be established on the oracle's side, by identity tests against the model's own reference implementation, and the harness documents this as a trust boundary rather than leaving it. Within those conditions, the enclosures are proofs, and the paper will call them that without further hedging.

2.3 Fail-closed beats fail-noisy

Existing numerical practice, where it acknowledges error at all, fails *noisily*: condition-number warnings, convergence flags, NaN propagation, significance stars on a diagnostic page. Noise is advisory, and advisory signals are ignored under deadline, buried by pipelines, and invisible by the time a number reaches a table. The decision embodied in Eq. (2) is to fail *closed*: a computation that cannot prove its claim does not proceed, and there is no flag to miss because there is no result to carry it. Three concrete harness behaviors illustrate the policy. The rigorous integrator refuses, rather than returns, any result that is non-finite or wider than the caller's stated tolerance.

The dual-path tripwire, which runs a cheap route and an independent oracle route to the same quantity, halts with both values on any disagreement, and is forbidden by construction from averaging them. Monte Carlo helpers refuse out-of-domain requests that formerly hung forever or returned silent NaN pairs. In each case the fail-noisy alternative was not imagined; it was measured in this package's own components before the contract was imposed, and Section 4.3 reports those measurements as results.

The policy has a cost. A fail-closed harness refuses some requests that a float pipeline would have completed correctly by luck, and it makes the caller do work (raise precision, shrink a box, repair an oracle) that a warning would have let them skip. The position of this paper is that for numbers that matter the trade is obviously right, and that the reason it has not been standard practice is architectural: until the refusal carries a receipt that says what to fix, refusing is merely obstruction. The receipt makes fail-closed livable.

3 The harness

`baller` is organized as five modules behind one integrity layer, totaling just under five thousand lines: deliberately small, because every line is inside the trust perimeter of a proof. The architecture has four layers, described here from the top: the modules and their contracts, the integrity layer that decides whether the package will run at all, the hygiene instruments that police the recurring traps of multi-precision code, and the structural instrument that sets the harness's reach, the block-arrow Krawczyk step.

3.1 Five modules and their contracts

Table 1 lists the modules with the contract each enforces and the refusal each can issue. Two design rules cut across all five. First, components are never trusted because of where they came from: engines adopted from earlier production use are carried as vendored source with content-hash pins and load through the integrity layer like everything else (Section 3.2). Second, policy knobs live with the caller, never inside the certifying code: the local-minimum instrument, for example, accepts a radius policy but does not own one, so that no tolerance buried in the harness can quietly widen what a certificate means.

The module boundaries also record provenance. The transport engines and the interval-Newton machinery were extracted from two earlier production pipelines (validated period transport for string-vacuum computations, and an exactly-scored forecast evaluator), where they had already survived production loads; the tripwire, the mutation harness, the hygiene instruments, and the block-arrow instrument were written for `baller`. The assembly is one discipline over both kinds: the same pins, the same battery, the same refusal semantics, whatever the origin of the bytes.

3.2 Integrity: verification before every run

A harness whose outputs are proofs must take seriously the question of what, exactly, is running. `baller` answers it fail-closed, like everything else. A verification entry point, run first in every consuming process, checks every vendored engine against pinned content hashes and refuses, typed, on any mismatch. Engine loads compile from source bytes with the interpreter's bytecode cache bypassed and purged, so a poisoned cache file cannot substitute for pinned source; loads are serialized under a lock, and the interpreter's module table is restored around every load so that a

module	provides	refuses
`quad`	rigorous integration on the Arb integrator [18] behind a fail-closed reader; positive-quadrature evidence for kernel functions	non-finite result; radius above the caller's cap (`BallBlown`)
`transport`	validated transport of ordinary-differential-equation solutions: exact recurrence towers, majorant tail bounds, arbitrary-precision Taylor steps with proven step-size contracts	step outside its proven contract ($\lvert h\rvert < r$ not provable, non-positive radius, wrong state dimension)
`certify`	Krawczyk and interval-Newton proof of existence and uniqueness of zeros; the block-arrow instrument of Section 3.4 with its interval positive-definiteness leg	contraction not provable (named block or border, measured margin); violated oracle contract, typed
`contract`	halt-not-average discipline: dual-path tripwire against an independent oracle; mutation-tested check harness; exact binomial (Clopper–Pearson) bounds for Monte Carlo quantiles	any dual-path disagreement (both values reported); out-of-domain requests; checks whose mutants pass
`hygiene`	static checks for the four recurring multi-precision traps (Section 3.3); radius watches and a re-round-safe precision bridge at run time	radius past the caller's relative bar; non-finite midpoints, including the infinite ball whose radius is zero

Table 1: The five modules. Every "refuses" entry is a typed exception with a receipt, per Eq. (2); none is a warning. The reach of the package is set by `certify`; the credibility of all five is set by the battery of Section 4.1.

component's import cannot leave a shadow for the next one. Components adopted by reference are authenticated by import-spec provenance and on-disk existence rather than by name alone, which defeats accidental shadowing and naive plants. Missing files, unreadable files, named pipes, and symlinked stand-ins each draw a typed refusal rather than a hang.

The integrity layer's boundary is this: an adversary already executing inside the process, able to forge the interpreter's own module machinery, can defeat any in-process check by the same means it would use to modify the harness itself. The integrity layer is engineered against the failure classes that actually occur (drifted files, stale caches, colliding module names, concurrent loads), each of which was demonstrated against the package during the break rounds of Section 4.2, and it documents the in-process limit as a boundary, not a closed it.

3.3 Hygiene: standing instruments for the recurring traps

Multi-precision code has a small set of traps that recur across every project that touches it, and `baller`'s position is that recurring traps deserve standing instruments rather than folklore. Four static checks scan source for the classes with the worst track record: precision fixed at import time rather than at call time; constants converted at whatever ambient precision happens to be in force; caches keyed without the precision that produced their contents; and precision context changed without restoration. Each check carries fixture cases measured from real code, and a finding is a nonzero exit, not a note. At run time, a radius watch halts a computation the moment a tracked ball's relative radius crosses the caller's bar, rather than letting a degraded enclosure flow to the end and fail there; midpoint trimming is permitted only after accepted steps, never inside

a trial step; and a dedicated bridge moves values between the harness and `mpmath` [19] without re-rounding, because both the obvious conversion and the officially recommended tuple round-trip silently re-round at the destination precision, an effect measured at 115 bits of loss on the trap path that motivated the bridge. These are advisory instruments with measured fixtures, and the paper is explicit about their epistemic weight: a clean hygiene pass certifies nothing. The checks exist to catch the traps that have actually been stepped in, and their fixture files are the record of the stepping.

3.4 Local minima at scale: the block-arrow Krawczyk step

The instrument that sets `baller`'s reach answers a question every fitted model raises and almost no pipeline can answer: the optimizer stopped, so is there actually an optimum here? Formally, given a candidate $\tilde{x} \in \mathbb{R}^n$ for a stationary point of an objective with gradient F and Hessian H, the harness seeks a box $X = \tilde{x} + [-r, r]^n$ (with r a vector of per-coordinate radii) in which a true zero of F provably exists, is provably unique, and H is provably positive definite throughout: a strict local minimum, with certificate. The classical tool is the Krawczyk operator [7, 9],

$$K(X) \;=\; \tilde{x} \;-\; C\,F(\tilde{x}) \;+\; \big(I - C\,H(X)\big)\,\big(X - \tilde{x}\big), \tag{3}$$

where C is any approximate inverse of the midpoint Hessian and $H(X)$ is an interval enclosure of the Hessian over the whole box: if $K(X)$ lands strictly inside X, a zero of F exists and is unique in X. The test is run per coordinate, in certified arithmetic, as $\mathrm{mag}(c_k) + (|E|\,r)_k < r_k$ with $c = CF(\tilde{x})$ and $E = I - CH(X)$; a comparison the arithmetic cannot decide is a refusal, per the contract. The float preconditioner C is heuristic and the rigor never depends on it: a bad C can only cause a refusal, not a false certificate.

Eq. (3) becomes an instrument, rather than a textbook entry, through structure. Hierarchical models, the kind that dominate applied statistics, have Hessians with an arrow shape: B independent diagonal blocks D_i, one per latent group, coupled only through a shared border of g global parameters,

$$H \;=\; \begin{pmatrix} D_1 & & & B_1 \\ & \ddots & & \vdots \\ & & D_B & B_B \\ B_1^\top & \cdots & B_B^\top & G \end{pmatrix}, \tag{4}$$

and a dense treatment of Eq. (3) at n in the thousands is hopeless in interval arithmetic. The harness assembles the Krawczyk test blockwise: an approximate inverse per diagonal block, an interval Schur complement on the border, and cross-block row sums bounded through factored intermediates so that the cost of the coupling terms is linear in the number of blocks rather than quadratic. Refusals from the block-arrow test name the failing coordinate block or the border, with the measured contraction factor, so that a refusal at five thousand coordinates is as actionable as one at five. Figure 1 shows the shape and the scale at which it runs in production: n of 5,592 and 5,955 with roughly six hundred blocks, proven in about 90 seconds.

Positive definiteness gets its own structured leg, and its design carries a measured lesson. The tempting route, certifying each diagonal block and then the border's Schur complement with per-block inverse bounds, fails in practice exactly where it is needed: on nearly collinear latent blocks with condition numbers near 10^{13}, a Neumann-series bound collapses to claiming nothing, and a certified interval solve inflates its enclosure by two orders of magnitude. The leg that survives is one global congruence: take the floating-point block-arrow Cholesky factor of the

midpoint matrix, equilibrate by exact powers of two, and certify in interval arithmetic that the congruence of the enclosure by that exactly-representable, exactly-nonsingular triangular factor is positive definite, with no rigorous inverse computed anywhere. The float factor, like C above, is a heuristic that can only cause refusal. Section 5.3 tells the story of how this leg reached its final form: in production, on a live model, inside one day.

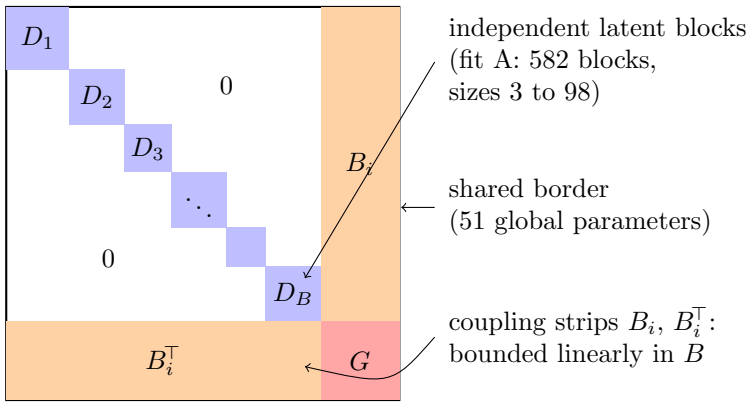

Figure 1: The block-arrow Hessian, Eq. (4), of a hierarchical objective: one diagonal block per latent group, all coupling through the border of shared parameters. `baller` runs the Krawczyk test of Eq. (3) and the positive-definiteness congruence in this structure, never densely, with the cross-block error terms bounded at cost linear in the number of blocks. The labels give the production scale of Section 5: 5,592 coordinates in 582 blocks of size 3 to 98 plus a 51-parameter border.

4 Hardening as content

The hardening record stands here as a result in its own right, with the same standing as the production record. The record has three layers: a battery in which every check is itself checked, adversarial break rounds run by fresh attackers, and a catalogue of measured traps.

4.1 The battery: checks that are themselves checked

The package carries a sixteen-leg acceptance battery, run from a scratch directory, that passes end to end in 41 seconds. Its organizing principle is mutation testing: a check earns trust only by demonstrably failing when its subject is broken, so battery legs systematically feed the checks tampered pins, poisoned bytecode caches, foreign module shadows, sign-flipped mathematics, and starved precision, and require the typed refusal in each case. Table 2 summarizes the legs. The battery caught its own author twice, and both catches are in the table. The precision bridge of Section 3.3 was itself, as first written, the re-rounding trap it was built to prevent (the officially recommended tuple form re-rounds too; the 115-bit measurement above is from the battery's own fixture). And the rigorous-integration leg found that constants built at ambient precision poison an integrand at the seventeen-digit level, with the ball radius reporting the poisoning; the fixture that proves it stays in the battery as a negative control.

legs	what they prove
integrity (pins, tamper, caches, shadows)	bit-flipped vendored source, poisoned bytecode caches, and planted module shadows each draw the typed refusal; clean loads verify
component identity	components adopted by reference resolve to the authentic modules, with provenance authenticated
transport truth cases	a validated ODE step reproduces 256-bit reference values; three mathematical mutations of the engine all fail the case, including the tail-sensitive one
quadrature evidence	the positive-quadrature check re-runs its adaptation battery live against a recorded seed (32 s)
exact statistics	Monte Carlo quantile bounds agree with an independent exact binomial computation to 10^{-12}
halt semantics	the tripwire halts on disagreement with both values; ball and complex inputs draw typed rejections; the mutation harness passes its own self-test
hygiene fixtures	each static check fires on its measured fixture and stays silent on fixed code
break-round regression locks (3 legs)	every fix from the adversarial rounds of Section 4.2 stays in force
rigorous integration	π to 53 certified digits; a boundary-layer integrand with a pole 2.8×10^{-17} off the contour to 49.7 certified digits in 0.01 s; the unflagged-singularity negative control stays wrong
block-arrow proofs	a planted zero is proven and contained; displaced and over-wide boxes draw named refusals; two oracle-mathematics sign flips are caught (one by the Krawczyk leg, one by the positive-definiteness leg); precision starvation refuses; a planted saddle is named; the scalar cross-check agrees

Table 2: The sixteen-leg battery, grouped by function; the ten rows group the sixteen legs of the battery manifest, and all sixteen pass end to end, run from a scratch working directory against the installed package, in 41 seconds. The design rule is that no check is trusted until a mutation of its subject has been observed to fail it. Legs marked as negative controls keep known-wrong behavior in the record so that a future "fix" cannot silently reintroduce it.

4.2 Three break rounds

After the battery first passed, the package was handed to adversarial reviewers with no stake in its success, in three rounds: three independent attackers briefed on integrity, evasion of the harness's own checks, and numerical correctness; then two more attacking the fixes and the uncovered surface; then a confirmatory round re-running every prior attack against the fixed code. The first two rounds produced six blocking and roughly two dozen major findings; every one was fixed, and the fixes were locked into the battery as three permanent regression legs, with the two step-contract blockers refused by the transport gate itself. The confirmatory round found the blockers dead and contributed four residual fixes of its own. Three of the blocking findings follow in detail, because each is a whole class of silent failure that a fail-closed design must actually close, and none of the three would announce itself in a float pipeline.

The first attacker found a cache-bypass shadow in the integrity layer itself: after a component's first verified load, a fast path served it from an in-memory cache without re-registering the module

table or re-checking for shadows, so a module planted afterward under the component's name was silently wired into the next verified engine load. The demonstration produced wrong numbers from a fully pin-verified engine with zero warnings. The fix re-authenticates on every path, restores the module table around cache hits, and serializes loads; the demonstration is now a battery leg. The second class was subtler: the tripwire, whose whole job is comparing two routes to the same number, silently accepted a *ball* as input, and the conversion took the midpoint, discarding the radius. A comparison instrument that quietly throws away the error bound of its input is worse than none; balls and complex values now draw typed rejections, with the rule that a radius may never be discarded by a scalar comparison. The third lived in the validated ODE step: driven outside its proven contract (a trial step larger than the proven radius), the engine silently took absolute values, converting a negative tail bound into a certified radius that was too small, which is the one direction a validated method must never err. The step now proves $|h| < r$ before moving, and refuses otherwise. All three failures were silent, and all three fixes are architectural: a path on which the silent case cannot proceed.

4.3 What the instrument refuses: measured negatives

The traps in Table 3 were each measured on this package's own code or inputs, at the magnitudes shown, before the corresponding refusal or contract existed. They are presented as results because each one is a quantitative statement about how multi-precision computing actually fails, portable to any project using the same substrates, whether or not it adopts this harness.

5 Production

`baller` has exactly one production consumer as of this writing, and that deployment is reported here in full, because its first day exercised every part of the design: the proof machinery, the refusal semantics, and the discipline for changing a certifying instrument while it is in use.

5.1 One consumer: `clinch`

`clinch` (Certified Local Interval-Newton Convergence for Hierarchies) is an instrument for fitted hierarchical models, built directly on `baller`'s block-arrow module: given a fitted optimum and model oracles evaluable in ball arithmetic, it either proves a strict local minimum in an explicit box around the candidate or refuses with the failure named. Its first production targets were two maximum-a-posteriori fits of a hierarchical Bayesian model of tropical-forest demography, with per-family latent blocks bordered by shared globals: 5,592 free coordinates for fit A (582 latent blocks plus a 51-parameter border) and 5,955 for fit B (627 blocks, same border). `clinch` owns the model oracle, the identity tests that establish oracle truth against the model's reference implementation (the trust boundary of Section 2.2), and the radius policy; `baller` owns the proof. The division matters for reading Table 4: every number in it is a statement about the model's own objective, anchored to the reference implementation to better than 5×10^{-10} relative before any proof was attempted.

5.2 The first refusal

The instructive event of the deployment is the first row block of Table 4. Fit A arrived with every sign of health a float pipeline can produce: the optimizer had stopped, and the largest entry of the gradient was 9.6×10^{-5}. The harness refused to call it an optimum, and the receipt

trap, as measured	magnitude	harness response
re-rounding on precision bridges: both the direct conversion and the recommended tuple round-trip re-round at destination precision	115 bits lost on the motivating path	dedicated raw-mantissa bridge; both lossy forms trapped by fixture
power function on a zero-containing ball returns NaN (routes through exp log)	total (NaN), silent until something reads it	non-finite results refuse; the coding rule ($x \cdot x$, never x^2 over a box) is enforced by fixture
unflagged endpoint singularity handed to the rigorous integrator	confidently wrong ball	kept as a live negative control; the analyticity flag is part of the calling contract
constants constructed at ambient precision inside a high-precision computation	enclosure degraded to the 17-digit class, radius honest	hygiene check plus battery fixture
uniform box radii on a stiff hierarchical Hessian (six decades of curvature)	width amplification defeats the test by 7.7×10^6	per-coordinate radius contract; the same candidate then proves
per-block inverse bounds for positive definiteness on nearly collinear blocks (condition $\sim 10^{13}$)	Neumann bound collapses to no claim; certified interval solve inflates $\sim 10^2$	single global congruence leg (Section 3.4)
comparison tripwire computing in double precision	disagreements past 16 digits invisible	comparisons at the bar plus 20 digits, in arbitrary precision
the infinite ball has radius zero, so a naive radius check reads it as exact	total misread of a non-finite state	non-finite midpoints halt before any radius shortcut

Table 3: Measured traps and the refusals they produced. Every magnitude was measured on this package's own components or production inputs. Read the table as the empirical case for Eq. (2): each row is a path on which a number would have gone wrong with no warning.

said why. In the metric that matters for a proof, the Newton step in the model's own curvature, the residual was 0.029 curvature standard deviations (written σ below), four orders of magnitude too loose to certify (the contraction factor stays near 2.9×10^4 even on a $10^{-6}\,\sigma$ box); enough to cover the true stationary point from that center forces Hessian variation so large that the contraction factor of Eq. (3) reaches 4.7×10^{10} on the border block, and no radius on the ladder wins. The refusal is correct: "the optimizer converged" at the fit's own tolerance was not a provable statement about this model. Two structured Newton steps, driven by the same block-arrow machinery, moved the center by 4.7×10^{-5} (maximum over coordinates) and brought the residual to 5.0×10^{-11} curvature units, and at that center the full proof went through in about 90 seconds. The deliverable is deliberately two-part: the proof at the polished center, plus the measured distance from the published point, so that a consumer of the original fit knows exactly how far the provable optimum sits from the numbers in their tables. Fit B repeated the pattern at smaller scale, one polish step from its reported point. Each refusal carried the number that predicted what would fix it, the receipt discipline of Section 2.2 at work.

5.3 The same-day amend cycle

Production also changed the instrument, three times, inside its first day; changing a proving instrument under load is exactly the situation its discipline is for. First, the original radius

	fit A	fit B
free coordinates	5,592	5,955
latent blocks + border	$582 + 51$	$627 + 51$
reported optimum: residual in curvature units	$0.029\,\sigma$	$2.0 \times 10^{-4}\,\sigma$
verdict on the reported point	refused (border, 4.7×10^{10})	refused (border, 3.3×10^{5})
Newton polish: steps, distance moved	2, 4.7×10^{-5}	1, 1.9×10^{-6}
residual after polish	$5.0 \times 10^{-11}\,\sigma$	$5.6 \times 10^{-11}\,\sigma$
verdict at the polished center	proven strict local minimum	proven strict local minimum
contraction factor (must be < 1)	0.735	0.834
positive-definiteness margin	≥ 0.98	≥ 0.97
per-coordinate radii	$[1.7 \times 10^{-13}, 10^{-9}]$	$[1.7 \times 10^{-13}, 10^{-9}]$
precision; wall time	192 bits; ~ 90 s	192 bits; ~ 90 s

Table 4: The first production proofs, both delivered on the instrument's first day in service. Both fits looked converged by their own tolerances, both were refused at the reported point with the border named, and both were proven strict local minima after a Newton polish costing less than 5×10^{-5} in parameter distance. Enclosure endpoints and margins are rounded outward from the receipts. The two identical bottom rows are shared policy, not copied entries: the working precision and the radius unit are code-pinned constants of the engine, and both polished residuals sit below the pinned radius floor, so both certificates ran at the same radius unit; the receipts' smallest per-coordinate radii differ (1.78×10^{-13} for fit A, 1.755×10^{-13} for fit B) and coincide only under the rounding. About 90 seconds is the measured wall time of each certificate.

contract took one scalar radius for all coordinates; on a Hessian whose curvature spans six decades, any uniform box fails the width-amplification arithmetic by a measured factor of 7.7×10^{6}, so the contract gained per-coordinate radii, and the battery gained the stiff fixture on which the uniform box refuses while the anisotropic box proves the same candidate. Second, the positive-definiteness leg was rebuilt from per-block bounds to the single global congruence, after both per-block routes failed measurably on fit A's nearly collinear blocks (the collapse and inflation quoted in Table 3). Third, the refusal register (the component that assembles refusal receipts) was tightened: radius information now passes through the oracle evaluation, and every margin is keyed to whether its row was certified. The sixteen-leg battery was re-run to a full pass after each of the three edits, so an unverified version was never the version in use. Maintenance for this class of software worked as intended: a production failure becomes a measured negative, then a contract change and a permanent battery leg, within hours and without a gap in coverage.

6 The exhibit program

Two public exhibits are registered for the harness, with designs fixed before their results; stating the designs in advance commits the eventual numbers to be judged against a fixed target. No result from either exhibit appears in this paper, and nothing below should be read as a claim about how they will come out.

6.1 The NIST reference problems

The first exhibit runs `baller` over the NIST Statistical Reference Datasets [2]: the linear-regression, analysis-of-variance, univariate, and nonlinear-regression suites whose reference values

exist precisely because production packages historically disagreed with them [3, 4]. The design is the contract of Eq. (2) applied to each problem: for every dataset, either a proven enclosure of each reference quantity, with certified digits reported, or a typed refusal with its receipt. The outcome space is stated in advance: on problems constructed to be ill-conditioned, a correct instrument may refuse at a given precision, and a refusal whose receipt names the conditioning is a pass for the semantics, whatever it is for the precision budget. The comparison of interest maps the failure surface when every answer must be proven or declined: which problems on the McCullough-era failure lists become enclosures, at what precision and cost, and which draw refusals, with what named cause. Beating double-precision packages on the easy suites would prove nothing and is no part of the design.

6.2 The trap suite

The second exhibit turns Table 3 outward: each measured trap, recast so it can be presented to other interval systems, run against raw python-flint [14], the IEEE 1788 decoration system as implemented in conforming libraries [16, 17], and the R interface to the same substrate [15]. For each trap and system the exhibit records one of four outcomes: a silently wrong or silently degraded result; a non-finite or vacuous result with no signal; an advisory signal (a decoration, a flag) attached to a still-flowing value; or a halt with an actionable record. The design premise, argued in Section 7, is that the fourth column is essentially empty in public software today; the exhibit exists to test that premise mechanically and to publish the grid either way. Both exhibits run in minutes by construction, so their eventual publication carries no resource caveat.

7 What exists, and what was missing

The claim of this paper is a layer claim, resting on an accurate account of the substrates; that account comes first.

7.1 The substrate landscape

Nothing in `baller` replaces them. Arb [11], now part of FLINT [12], is the reference ball-arithmetic substrate: arbitrary precision, rigorously bounded special functions, certified linear algebra, and rigorous integration [18], engineered and documented to a standard this paper relies on throughout. python-flint exposes it to Python [14], and the recent CRAN package brings the same substrate to R with native S4 semantics [15], which means ball arithmetic as such is now available in the daily language of applied statistics. INTLAB [13] has provided verified numerics in MATLAB since 1999, including verified nonlinear-system solving in the dense, double-precision setting, and remains the reference point for what a mature verification toolbox looks like. IntervalArithmetic.jl gives Julia a conforming, actively maintained interval foundation [16]. Above all of these, the IEEE 1788-2015 standard [17] defines decorations: per-result flags that record whether an operation left its domain of validity, the standardized acknowledgment that an interval result needs provenance attached. The theory these systems implement [6–10] is mature and not in question.

7.2 The empty layer

None of these systems supplies the policy layer of Eq. (2). In every one of them, the wide ball, the NaN, the undecorated or badly decorated interval is returned to the caller, and everything

after that point keeps computing with it by default; decorations, the closest existing mechanism, are advisory by design and carry no receipt beyond a flag. To our knowledge, no public library in any of these ecosystems makes enclosure-or-typed-refusal the contract of every operation, attaches machine-readable receipts to refusals, mutation-tests its own checking apparatus, or provides a block-arrow Krawczyk instrument at the block-arrow scale demonstrated in Section 5; the prior-art search for the structured instrument, run before it was built, found dense-setting local verification but nothing at hierarchical structure and scale. `baller` occupies that layer. Ball arithmetic in a scripting language, to be clear, is old news, and the CRAN package alone would refute any claim of novelty there. The contribution is the semantics bolted on top of the substrate; the receipts, which make refusals actionable; the hardening record behind the checks; and the block-arrow instrument, which makes proofs at fitted-model scale routine.

8 Outlook

The harness is in production use; the remaining work puts it in more hands and in front of more problem classes, and both motions are under way.

8.1 Availability and the near term

`baller` is at present an internal instrument: the version documented here is the version in production use, and its public release, under this name, is being prepared, with the exhibit program of Section 6 to be published against it. Until release, this paper is the package's public specification, and it has been written so that the design can be judged, and independently reimplemented, from the paper alone: the contract of Eq. (2), the structured test of Eqs. (3) and (4), the battery design of Table 2, and the trap catalogue of Table 3 carry the design, and none of them depends on access to the code.

8.2 Where refusal semantics goes next

The block-arrow instrument applies wherever a fitted model has latent groups bordered by shared parameters, which is to say across most of hierarchical statistics: mixed models, random-effects meta-analysis, multilevel regression, the standard apparatus of ecology, education research, and epidemiology. Any such model whose gradient and Hessian can be evaluated in ball arithmetic is a candidate for the same proof, and the consumer of Section 5 is evidence that the oracle work takes days. The wider pattern is not specific to optimization. Any computation with a checkable success criterion can be run under Eq. (2), and the components here (the tripwire, the exact quantile bounds, the hygiene instruments) were built as the general kit for doing so. Fifty years of numerical practice treat the printed number as innocent until proven guilty; everything reported in this paper came from running the presumption the other way. Wherever a theory meets data through software whose answers must be right, the fail-closed default is now cheap enough to be the default in fact.

Acknowledgments

Funding for this project was provided by Anthropic. This is not an official Anthropic project and results and views are not endorsed by Anthropic.

References

[1] James W. Longley. An appraisal of least squares programs for the electronic computer from the point of view of the user. *Journal of the American Statistical Association*, 62(319): 819–841, 1967. doi: 10.1080/01621459.1967.10500896.

[2] National Institute of Standards and Technology. Statistical Reference Datasets (StRD). https://www.itl.nist.gov/div898/strd/, 1998.

[3] B. D. McCullough. Assessing the reliability of statistical software: Part I. *The American Statistician*, 52(4):358–366, 1998. doi: 10.1080/00031305.1998.10480597.

[4] B. D. McCullough. Assessing the reliability of statistical software: Part II. *The American Statistician*, 53(2):149–159, 1999. doi: 10.1080/00031305.1999.10474450.

[5] B. D. McCullough and H. D. Vinod. The numerical reliability of econometric software. *Journal of Economic Literature*, 37(2):633–665, 1999. doi: 10.1257/jel.37.2.633.

[6] Ramon E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[7] Rudolf Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969. doi: 10.1007/BF02234767.

[8] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[9] Siegfried M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010. doi: 10.1017/S096249291000005X.

[10] Warwick Tucker. *Validated Numerics: A Short Introduction to Rigorous Computations*. Princeton University Press, Princeton, NJ, 2011.

[11] Fredrik Johansson. Arb: Efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66(8):1281–1292, 2017. doi: 10.1109/TC.2017.2690633.

[12] The FLINT team. FLINT: Fast library for number theory. https://flintlib.org, 2025.

[13] Siegfried M. Rump. INTLAB – INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[14] The python-flint developers. python-flint: Python bindings for FLINT. https://github.com/flintlib/python-flint, 2025.

[15] Mikael Jagan. flint: Fast library for number theory. R package version 0.1.5, https://CRAN.R-project.org/package=flint, 2025.

[16] Luis Benet and David P. Sanders. IntervalArithmetic.jl: Library for validated numerics using interval arithmetic. https://github.com/JuliaIntervals/IntervalArithmetic.jl, 2025.

[17] IEEE. IEEE standard for interval arithmetic. IEEE Std 1788-2015, 2015.

[18] Fredrik Johansson. Numerical integration in arbitrary-precision ball arithmetic. In *Mathematical Software – ICMS 2018*, volume 10931 of *Lecture Notes in Computer Science*, pages 255–263. Springer, 2018. arXiv:1802.07942.

[19] Fredrik Johansson et al. mpmath: a Python library for arbitrary-precision floating-point arithmetic. https://mpmath.org, 2023.