SSE PROOF SERIES — STRUCTURAL GOVERNANCE IN ACTION

SHUNYAYA STRUCTURAL EQUATIONS (SSE)
Status: Public Research Release (v1.3)
Date: January 15, 2026
Caution: Research and observation only. Not for critical or automated decision-making.

EXECUTIVE SUMMARY — SSE PROOF SERIES

The Shunyaya Structural Equations (SSE) framework introduces a governing capability that classical mathematics does not possess:

the ability to determine whether mathematically correct results are responsible to rely on.

This document exists for one purpose:

to demonstrate, using executable and reproducible evidence, that structural trust governance is real, deterministic, and operational on real mathematics and real data.

Unlike the main SSE document—which defines axioms, canonical forms, and governing principles—this Proof Series is strictly empirical and operational.

It focuses exclusively on:

- real-world datasets or declared classical regimes
- replayable executions
- deterministic structural decisions
- explicit **ALLOW**, **DENY**, and **ABSTAIN** outcomes

No simulations are used.
No synthetic data is introduced.
No classical mathematics is altered.

Every numerical value produced in this document collapses exactly to its classical counterpart under the invariant:

```
phi((y, a, s)) = y
```

ROLE OF THIS DOCUMENT IN THE SSE ECOSYSTEM

The SSE framework is intentionally published as two separate and complementary documents:

Document 1 — SSE Axioms (Foundational)
Defines what SSE is, the axioms it obeys, and the principles governing admissibility, denial, abstention, and trust continuity.

Document 2 — SSE Proof Series (This Document)
Demonstrates that those axioms operate correctly in practice, on executable mathematics, without modifying equations or solvers.

This separation is deliberate.

Axioms define responsibility.
Proof demonstrates credibility.

No axiomatic claim in SSE stands without executable evidence.

WHAT THIS DOCUMENT PROVES — AND WHAT IT DOES NOT

This document proves that:

- **SSE abstains** when mathematics is structurally undefined
- **SSE denies reliance** before numerical catastrophe occurs
- **SSE allows and concludes safe convergence** under admissible conditions
- **SSE decisions are deterministic, reproducible, and replayable**
- **SSE governance operates without modifying equations, solvers, or results**

This document does **not** attempt to prove:

- improved numerical accuracy
- better optimization performance
- faster convergence
- predictive superiority

SSE is **not** a performance framework.

SSE is a trust-governance framework.

PROOF METHODOLOGY

Each proof case in this document follows the same strict structure:

1. A real dataset or a declared classical mathematical regime is selected
2. A standard classical method is applied **unchanged**
3. SSE governance is layered externally
4. Classical and SSE traces are recorded side by side
5. Outcomes are compared **only at the trust level**, never at the value level

Each case produces one or more of the following **canonical outcomes**:

- **ALLOW**
- **CONVERGED_ALLOW**
- **DENY**
- **ABSTAIN**

These outcomes are not heuristic.
They follow directly and exclusively from SSE axioms.

WHY REPLAYABLE FAILURES MATTER

Most real-world failures are not caused by incorrect mathematics.

They occur because:

- equations are applied outside safe regimes
- solvers are trusted beyond structural stability
- extrapolations are relied upon without admissibility checks

Post-incident analyses often conclude:

“The math was correct, but the conditions were unexpected.”

SSE exists to formalize this missing layer.

The proof cases in this document are **failure replays**—
demonstrating that SSE would have **denied reliance or abstained** *before* damage occurred,
without altering a single classical computation.

PROOF SERIES OVERVIEW

This document currently contains the following **verified and executable proof cases**:

Case 1 — Numerical Solver Failure Replay (MGH17)

Demonstrates abstention, denial, and safe allowance in a real nonlinear least-squares solver using a public benchmark dataset.

Shows that mathematically correct numerical procedures may become structurally inadmissible under singularity, overflow, or instability—**without altering the solver itself**.

Case 2 — Calculus Linearization Denial Near Instability

Demonstrates that calculus linearization remains mathematically correct while becoming structurally inadmissible near unstable regimes.

Using deterministic execution on classical functions with known instability boundaries, this case shows:

- **ALLOW** in stable regions
- **DENY** near reproducible structural boundaries
- **ABSTAIN** where derivatives or expressions are undefined

Calculus is not modified.
Only trust is governed.

Planned Extensions (Designed, Not Speculative)

- **Case 3 — Competing Model Selection Under Structural Conflict**
- **Case 4 — Threshold Crossing and Boundary Misuse**
- **Case 5 — Structural Abstention in Ambiguous Regimes**

Each case stands alone and is independently reproducible.

RELATIONSHIP TO THE SSE AXIOMS

All proof cases in this document are direct applications of the SSE axioms, most centrally:

SSE Axiom 0 — Structural Admissibility

An equation may be mathematically correct and yet be structurally inadmissible.
No equation is permitted to claim trust at a point unless its structural state is admissible at that point.

The proofs do not extend the axioms.
They **demonstrate** them.

REPRODUCIBILITY AND TRANSPARENCY

For each proof case, the following artifacts are produced:

- a dataset or declared mathematical regime
- a classical computation trace
- an SSE structural trace
- explicit per-iteration trust outcomes

No hidden parameters are used.
No adaptive learning is involved.
All thresholds are explicit and documented.

Independent reproduction is not optional. It is expected.

TRANSITION TO PROOF CASES

What follows begins the formal SSE Proof Series.

The first case establishes the minimum capability required for any governing framework:

the ability to prevent unsafe reliance on numerically unstable mathematics.

The second case extends this capability into calculus itself, demonstrating that even foundational mathematical tools require governance when approaching instability, boundaries, or undefined regimes.

CASE 1 — NUMERICAL SOLVER FAILURE REPLAY (MGH17)

Objective of This Case

The purpose of Case 1 is to demonstrate, on a real and established nonlinear regression dataset, that **Shunyaya Structural Equations (SSE)** can:

- **ABSTAIN** when mathematics is structurally undefined
- **DENY** reliance before numeric catastrophe
- **ALLOW** and **CONVERGED_ALLOW** safe convergence when conditions are admissible

All **without modifying the underlying mathematics or solver logic**.

This case establishes the **minimum credibility requirement** for SSE as a governing layer over numerical mathematics.

Dataset Overview

The dataset used in this case is **MGH17**, a canonical nonlinear least-squares benchmark.

Key characteristics:

- 33 real data points
- Widely used to test robustness of nonlinear solvers
- Known to exhibit singularities and numeric instability depending on initialization
- Includes documented failure modes and certified starting values

The model form is:

```
y = b1 + b2*exp(-b4*x) + b3*exp(-b5*x)
```

This dataset is intentionally difficult and is routinely used to expose solver pathologies that remain invisible until failure.

Classical Method (Unmodified)

The classical method applied is **Gauss–Newton nonlinear least squares**, implemented directly from first principles:

- Normal equations formed as
  `J^T * J * delta = J^T * r`
- Optional diagonal damping (when enabled)
- **No trust region**
- **No heuristics**
- **No fallback logic**

This is a deliberate choice.

SSE is evaluated on top of the classical method, not instead of it.

SSE Overlay

SSE introduces a **structural evaluation at each iteration** without altering the solver.

Each iteration is evaluated for:

- structural permission `a`

- accumulated structural resistance `s`
- conditioning posture
- step magnitude
- improvement posture

Each iteration results in exactly one of the following outcomes:

- **ALLOW**
- **CONVERGED_ALLOW**
- **DENY**
- **ABSTAIN**

The classical update rule remains unchanged.
Only reliance on the update is governed.

Case 1-A — Structural Undefinedness (Singular Normal Matrix)

Initial Condition
Start 1 (large-magnitude initialization)

Observed Classical Behavior

- Gauss–Newton encounters a singular normal matrix at the first iteration
- The solver step is mathematically undefined

Outcomes

- Classical status: **SINGULAR_JTJ**
- SSE status: **ABSTAIN_SINGULAR**

Interpretation

The equation remains mathematically correct in form, but **no structurally admissible step exists**.

- SSE does not attempt recovery
- SSE does not force progress
- SSE does not modify the solver

SSE abstains deterministically.

This demonstrates **formal abstention**, not failure.

Case 1-B — Numeric Catastrophe Prevention (Damped Solver)

Initial Condition
Start 1 with diagonal damping enabled

Observed Classical Behavior

- The classical solver proceeds briefly
- Numeric overflow and invalid values occur at iteration 2

Outcomes

- Classical status: **NUMERIC_FAIL**
- SSE status: **DENY** at iteration 1

Interpretation

The classical solver attempts to continue into a numerically unstable regime.

SSE denies reliance before catastrophic numeric failure occurs.

- No solver logic is changed
- No corrective action is taken
- No heuristic intervention is applied

This is a direct replay of a real-world solver failure scenario, governed deterministically.

Case 1-C — Safe Convergence Allowance

Initial Condition
Start 2 (moderate initialization, structurally admissible regime)

Observed Classical Behavior

- The classical solver executes the full iteration budget
- Convergence is gradual, stable, and numerically well-behaved

SSE Configuration

- Lenient early-regime allowance
- Strict collapse preservation
- Explicit structural convergence detection

Outcomes

- Classical status: **CLASSICAL_STEP** (50 iterations)
- SSE status: **CONVERGED_ALLOW** (6 iterations)

Interpretation

SSE permits normal solver behavior throughout admissible regions.

- SSE does not interfere with convergence
- SSE does not alter solver updates
- SSE concludes reliance once structural convergence is achieved

This demonstrates **responsible allowance**, not obstruction.

Summary of Case 1 Outcomes

Across three structurally distinct regimes, SSE demonstrates all governing outcomes:

- **ABSTAIN** when mathematics is structurally undefined
- **DENY** before numeric catastrophe
- **CONVERGED_ALLOW** under safe and stable conditions

At no point is the classical computation altered.

The distinction lies entirely in **when results are permitted to claim trust**.

Why This Case Matters

Most mathematical failures in real systems are not caused by incorrect equations.

They occur because:

- solvers are trusted beyond structurally safe regions
- numeric instability is detected too late
- misuse is recognized only after damage occurs

Case 1 demonstrates that SSE makes these failure modes explicit, deterministic, and preventable.

Reproducibility Notes

All results in this case are:

- deterministic
- reproducible
- based on real data
- traceable at each iteration

All Case 1 outcomes are generated from a single canonical dataset using identical classical code paths, differing only in structural admissibility conditions.

Artifacts produced per outcome:

- canonical dataset CSV
- classical computation trace (CSV)
- SSE structural trace (CSV)

Independent reproduction is straightforward and encouraged.

External Dataset License and Citation (Case 1)

Dataset: MGH17
Source: National Institute of Standards and Technology (NIST), Statistical Reference Datasets (StRD) — Nonlinear Least Squares Regression
URL: https://www.itl.nist.gov/div898/strd/nls/data/mgh17.shtml

The MGH17 dataset is provided by NIST, a U.S. Federal Government agency.
As a U.S. Government work, the dataset is treated as **public domain** under U.S. Federal Government policy.

Proper citation of the NIST source and URL is required when reproducing results.

Recommended citation:

National Institute of Standards and Technology.
MGH17 Dataset — Statistical Reference Datasets (StRD), Nonlinear Least Squares Regression.
https://www.itl.nist.gov/div898/strd/nls/data/mgh17.shtml

CASE 2 — CALCULUS LINEARIZATION DENIAL NEAR INSTABILITY (EXECUTABLE PROOF)

Objective

The objective of Case 2 is to demonstrate that **calculus may remain mathematically correct while becoming structurally inadmissible to rely on near unstable regimes**.

SSE does not replace calculus and does not modify classical outputs.
SSE governs whether a result may responsibly claim trust.

Setup

We test first-order Taylor linearization of a function at a point `x` with step `h`:

- True value: `y_true = f(x + h)`
- Linearized value: `y_lin = f(x) + f'(x) * h`
- Error: `err = abs(y_true - y_lin)`

We choose a function with a known instability boundary:

- `f(x) = sqrt(x)`
- Instability boundary at `x <= 0`

Classical calculus defines behavior wherever the function and derivatives exist.

SSE Overlay (Governance Only)

SSE introduces structural evaluation alongside the classical computation.

The classical output remains unchanged and the collapse invariant is preserved:

- `phi((y_lin, a, s)) = y_lin`

SSE computes:

- structural permission `a`
- cumulative structural resistance `s`

No calculus object is altered.

Deterministic Structural Signals

A deterministic risk index is derived from local sensitivity and curvature:

- `G = abs(f'(x))`
- `C = abs(f''(x))`
- `r = (C * abs(h)) / (1 + G)`
- `a = 1 / (1 + r)`

Structural resistance accumulates only when risk exceeds a declared safe limit:

- `s_{k+1} = s_k + max(0, r_k - r_safe)`

All quantities are deterministic and directly computable from classical derivatives.

Governance Rule

At each evaluation point, SSE returns exactly one status:

- **ALLOW** if `a >= a_min` and `s <= s_max`
- **DENY** if `a < a_min` or `s > s_max`
- **ABSTAIN** if the expression or derivative is undefined

Denial may occur even while calculus remains defined.

Declaration Used (Deterministic Parameters)

- `a_min = 0.70`
- `s_max = 0.80`
- `r_safe = 0.15`
- `h = 1e-3`

No tuning or learning is involved.

Execution (Deterministic, No Dataset Required)

Run from the SSE root:

```
python scripts\sse_case2_calculus_linearization.py --fn sqrt --root .
```

This generates three scenario folders, each containing:

- `trace_classical.csv`
- `trace_sse.csv`

Scenario Results (Canonical Outcomes)

1. Safe Corridor — ALLOW

A region far from the instability boundary (`x` comfortably above 0).

Observed behavior:

- `y_lin` closely matches `y_true`
- permission `a` remains high
- resistance `s` remains near zero
- SSE returns **ALLOW** throughout the corridor

Interpretation
Calculus linearization is structurally admissible.
SSE does not interfere.

---

2. Near-Boundary Corridor — DENY Transition

A corridor approaching the instability edge (very small positive `x`).

Observed behavior:

- calculus remains numerically defined
- permission `a` drops below `a_min` and/or resistance `s` exceeds `s_max`
- SSE transitions deterministically from **ALLOW** to **DENY**

Example corridor summary:

- `x_min ≈ 0.0009`, `x_max ≈ 0.0016`
- observed `min(a) ≈ 0.656` (below `a_min = 0.70`)
- observed `max(s) ≈ 1.90` (above `s_max = 0.80`)
- clean boundary: **ALLOW 4 / DENY 4**

Interpretation
Calculus remains mathematically correct, but **reliance becomes structurally inadmissible near instability**.

---

3. Instability Corridor — ABSTAIN

A corridor at or beyond the domain boundary (`x <= 0`).

Observed behavior:

- derivatives become undefined
- linearization produces NaN
- SSE returns **ABSTAIN** deterministically

Interpretation
SSE refuses trust when the calculus object itself is undefined at the declared boundary.

Conclusion

Case 2 proves that SSE is a **working, executable governance layer for calculus**:

- calculus remains intact and computes wherever defined
- collapse to classical meaning is preserved

- SSE deterministically governs trust:
    - **ALLOW** in stable corridors
    - **DENY** near unstable regimes
    - **ABSTAIN** at invalid boundaries

This is not theoretical.
It is fully reproducible, fully deterministic, and executable on any machine without external data.
