Structural Superposition Computing

A Hybrid Dynamical Framework: Theory, Hardware, Algorithms, and Validation

Yuval Fradkin

2026

Abstract

We define Structural Superposition Computing (SSC) as a hybrid dynamical system on a finite-dimensional state space, combining continuous relaxation dynamics with a discrete, physically realizable projection ("collapse") map. This paper makes no claim of asymptotic computational advantage, structural compression, or super-Turing power. Its goal is to give SSC a mathematically well-posed definition; to characterize when its trajectories exist, are unique, converge, and resist noise; and to specify the collapse operator as a concrete, bounded-cost element rather than an oracle.

We prove local exponential convergence under strong convexity at the correct rate $\mu = \lambda\_min(H)$, derive a noise-induced accuracy floor via an overdamped Langevin model with physically consistent units, and use the Kramers–Arrhenius escape law to make explicit the central limitation of the framework: the energy barriers that stabilize stored states also slow search on non-convex landscapes. Well-posedness is established rigorously under Carathéodory conditions for measurable inputs. Simulability within polynomial time is stated precisely, with hypotheses sufficient to close the step-count argument. A hardware architecture is defined and a benchmark methodology is fixed. Four representative algorithms —matched filtering, spectral inference, associative memory, and graph relaxation as a heuristic—are analyzed under an operational ESRC membership test. Numerical experiments reproduce the three core theoretical predictions; negative results are reported explicitly. The result is a compact, honest foundation on which realizable hardware and empirical application studies can be built without inheriting unproven performance claims.

1. Introduction and Scope

We define Structural Superposition Computing (SSC) as a class of hybrid analog dynamical systems: continuous gradient-like relaxation, punctuated by a discrete, hardware-level quantization step. The framework deliberately restricts its target. SSC is treated here not as a universal accelerator but as a hybrid dynamical system whose mathematical claims are those that standard dynamical-systems and stochastic-analysis tools can support.

Concretely, this paper deliberately avoids all of the following, each of which was a source of error in the earlier formulation:

- effective-WOPS metrics that multiply a physical update rate by an algorithmic speedup;
- the Margolus–Levitin bound used as a performance predictor rather than a ceiling;
- the Bekenstein bound as a basis for symbolic capacity;
- any claim of advantage over quantum computation;
- proofs of “structural compression”;
- exact solution of NP-hard problems, MaxCut included;
- super-Turing capability; and unmeasured performance numbers.

What remains is a framework whose value is architectural—latency, parallelism, and energy for embedded structured operators—and whose theoretical claims are correspondingly modest and provable. The paper is organized as follows. Sections 2–5 establish the mathematical model, well-posedness, and convergence theory. Sections 6–8 treat quantization, noise, and metastability. Section 9 gives the computational interpretation. Section 10 situates SSC relative to existing analog and hybrid

models. Section 11 states all limitations plainly. Section 12 fixes the experimental interface. Section 13 defines the structural computation model and advantage envelope. Section 14 closes the quantum analogy. Sections 15–16 cover hardware architecture and substrate candidates. Sections 17–18 define the governing equations, benchmark methodology, and minimal prototype. Sections 19–22 develop the algorithm theory and four representative use cases. Sections 23–26 present the numerical validation suite, analytical case study, and negative results. Section 27 consolidates open problems, and Section 28 concludes.

One boundary of scope deserves an explicit statement up front. This paper does not seek a new complexity class or a Shor-like separation. It establishes the mathematical framework and asks a narrower algorithmic question: whether structural relaxation can yield new within-P computational primitives or measurable physical advantages for structured, repeatedly reused operators. Section 14 closes the quantum analogy explicitly; Section 13 defines the resource model in which the legitimate frontier can be studied.

2. Finite-Dimensional State Model

We begin in finite dimensions and treat any continuous-field description as a subsequent limit, not a starting point. This avoids the topological difficulties of the original formulation, where homotopy "sectors" were assumed on the linear space $H^2(X)$, which is convex and hence contractible and connected.

The state is a vector $a(t) \in \mathbb{R}^n$, evolving under the controlled, noisy dynamics

$$\dot{a}(t) = -\nabla F(a(t)) + B\,u(t) + \xi(t) \qquad (1)$$

where:

- $F : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable potential (the structural functional);
- $u(t)$ is an external input entering through a fixed matrix B; $u(\cdot)$ is assumed measurable and locally bounded;
- $\xi(t)$ is a physical noise term, specified precisely in Section 7.

The gradient ∇F is taken with respect to the Euclidean inner product on $\mathbb{R}^n$. When we later invoke a specific hardware realization, a should be read as a vector of physical observables (node voltages, currents, phases, or amplitudes), and F as the energy-like function whose gradient the circuit physically descends.

Remark 2.1. The restriction to finite n is not a mere convenience. It is the regime in which any physical device operates, and it is the regime in which the accuracy floor of Section 7 and the metastability analysis of Section 8 have operational meaning. The infinite-dimensional field picture is recovered only formally, as $n \to \infty$ with a chosen spectral basis, and no claim in this paper depends on that limit.

3. Hybrid Flow–Jump Formulation

We adopt the standard hybrid-systems formalism, which keeps the flow and jump regimes cleanly separated by a flow set and a jump set.

Fix a scalar coherence function $C : \mathbb{R}^n \to \mathbb{R}$ and threshold $\Theta \in \mathbb{R}$, and define

$$\mathcal{M}_\Theta = \{\, a : C(a) < \Theta \,\} \quad \text{(flow set)},$$

$$\mathcal{J}_\Theta = \{\, a : C(a) \geq \Theta \,\} \quad \text{(jump set)}.$$

The hybrid system is

$$\dot{a} = -\nabla F(a) + Bu + \xi, \quad a \in \mathcal{M}, \qquad (2)$$

$$a^{+} = \Pi(a), \qquad a \in \mathscr{T}, \quad (3)$$

where $\Pi : \mathbb{R}^n \to \mathbb{R}^n$ is the collapse map.

Design Constraint 3.1 (Bounded collapse). Π must be a fixed, explicitly specified map computable in O(n) elementary physical operations. Π may not be defined as "select the global optimizer," since that hides the problem's hardness inside the operator. Admissible choices are component-wise quantization,

$$\Pi_i(a) = +1 \text{ if } a_i \geq h; \ -1 \text{ if } a_i \leq -h; \ a_i \text{ if } |a_i| < h, \quad (4)$$

or a local winner-take-all,

$$\Pi(a) = e_{k^*}, \quad k^* = \operatorname{argmax}_k a_k. \quad (5)$$

Each of these maps to a real circuit element (a bank of comparators / Schmitt triggers, or a WTA network) whose time and energy cost is measurable. Because Π is bounded, no computational work is smuggled into the jump; whatever advantage the system offers must come from the continuous dynamics (2), where it can be honestly analyzed.

Remark 3.2 (Zeno avoidance). To ensure trajectories are not trapped in an infinite cascade of jumps, we require that $\Pi(\mathscr{T}) \subseteq \mathscr{N}$ after a fixed dwell: immediately following a jump the state re-enters the flow set and flows for at least a minimum dwell time $\tau_d > 0$ before the next jump is admissible. This is the standard temporal regularization for hybrid systems and is trivially satisfied by (4)–(5) with a hysteresis band.

4. Well-Posedness

Before any convergence or performance statement, the dynamics must be shown to have solutions that exist and are unique. This is where the finite-dimensional, locally-Lipschitz setting pays off.

Theorem 4.1 (Existence and uniqueness of flow). Suppose $F \in C^2$ with ∇F locally Lipschitz in a, and $u(\cdot)$ is measurable and locally bounded. Then the right-hand side of (2) (with $\xi \equiv 0$) satisfies the Carathéodory conditions: it is measurable in t for each fixed a, and locally Lipschitz in a uniformly over compact time intervals. By the Carathéodory existence and uniqueness theorem, for any initial condition $a(0) \in \mathscr{N}$ there exists a unique maximal absolutely continuous solution on the interval up to the first hitting time of $\mathscr{T}$. If F is additionally coercive and radially unbounded, sublevel sets are compact and forward-invariant under gradient descent, ruling out finite-time blow-up inside $\mathscr{N}$.

Proof. The Carathéodory conditions hold because ∇F is continuous (hence measurable in t for every fixed a), and $Bu(t)$ is measurable in t and locally Lipschitz in a with a t-independent Lipschitz constant on any compact set. The Carathéodory existence and uniqueness theorem (see, e.g., Coddington–Levinson) then gives a unique maximal absolutely continuous solution. Forward invariance under coercivity follows from standard gradient-flow theory. □

Proposition 4.2 (Well-posed hybrid execution). Under Theorem 4.1, together with the dwell-time condition of Remark 3.2, a bounded collapse map Π, and the assumption that the flow (2) is forward complete on $\mathscr{N}$ under the admissible inputs (no finite-time blow-up inside $\mathscr{N}$), the full hybrid trajectory (flow then jump then flow …) is well-defined for all $t \geq 0$ and admits no Zeno behavior.

Remark 4.3 (Sufficient condition for forward completeness). If F is coercive and radially unbounded and $u(\cdot)$ is essentially bounded, then $\dot{F} = -\|\nabla F\|^2 + \nabla F^T B u \leq -\|\nabla F\|^2 + \|\nabla F\| \cdot \|B\| \cdot \|u\|_\infty$, which is strictly negative for $\|\nabla F\| > \|B\| \cdot \|u\|_\infty$. This implies forward completeness, so coercivity plus bounded inputs is a verifiable sufficient condition for Proposition 4.2.

Proof. Between jumps the flow is unique by Theorem 4.1; forward completeness (assumed) prevents finite-time blow-up inside $\mathscr{N}$, so each flow segment reaches $\mathscr{T}$ or continues indefinitely. Each jump is a single evaluation of the bounded map Π. The dwell time $\tau_d > 0$ lower-bounds the interval between

consecutive jumps, so only finitely many jumps occur on any finite horizon; concatenating the flow segments yields a unique global hybrid arc. □

Remark 4.4. These statements are deliberately elementary. Coercivity of F is an assumption that must be verified for each specific application; it is not automatically guaranteed by the bounded-cost property of Π.

5. Stability and Convergence

The convergence rate of the relaxation is governed by the smallest positive eigenvalue of the Hessian at the target, equivalently the strong-convexity modulus μ, not by the spectral gap $\lambda_2 - \lambda_1$.

5.1 The correct rate

Linearize the noise-free flow about an equilibrium a★ where ∇F(a★) = 0. With H = ∇²F(a★) and v = a − a★,

$$\dot{v} = -H\,v + o(\|v\|).$$

If H ≻ 0 with eigenvalues $0 < \lambda_1 \le \dots \le \lambda\_n$, then ‖v(t)‖ ≤ e^{−λ₁t} ‖v(0)‖ — the slowest mode decays at rate λ_1, not $\lambda_2 - \lambda_1$.

Theorem 5.1 (Local exponential convergence). Let F be μ-strongly convex and L-smooth on a convex neighborhood of its unique minimizer a★ (so μI ≤ ∇²F ≤ LI there). Then the noise-free flow (2) satisfies, for a(0) in that neighborhood,

$$\|a(t) - a^\bigstar\| \le e^{-\mu t}\,\|a(0) - a^\bigstar\|,$$

and the time to reach accuracy ε is

$$T_\varepsilon \le (1/\mu)\,\log(\|a(0) - a^\bigstar\| / \varepsilon). \qquad (6)$$

Proof. Let V(a) = ½‖a − a★‖². Along the flow, $\dot{V} = -\langle a - a^\bigstar, \nabla F(a)\rangle$. Strong convexity gives $\langle a - a^\bigstar, \nabla F(a) - \nabla F(a^\bigstar)\rangle \ge \mu\|a - a^\bigstar\|^2$, and ∇F(a★) = 0, so $\dot{V} \le -2\mu V$. Grönwall's inequality yields V(t) ≤ e^{−2μt}V(0), i.e. the stated bound. Solving e^{−μt}‖a(0) − a★‖ = ε for t gives (6). □

Remark 5.2. The dependence is on $\mu = \lambda_1$, the smallest eigenvalue — equivalently 1/κ-like behavior once conditioning is accounted for (see Section 18, where settling time is tied to the condition number of the coupling matrix). The gap $\lambda_2 - \lambda_1$ governs convergence only in the special case $\lambda_1 = 0$ forced by symmetry, which is not the generic situation and was not the situation defined originally.

5.2 Non-convex landscapes: no global claim

For non-convex F we make no global convergence claim. For non-convex F, trajectories may converge to local minima depending on their basin of attraction; no global convergence claim is made. The honest statements available are: (i) convergence to a strict local minimizer for almost every initial condition under the strict-saddle property (by the stable-manifold theorem, unstable equilibria are avoided for generic initializations); (ii) a success probability under randomized restarts; or (iii) an average solution-quality / approximation-ratio statement for a specified problem family. Any of these is a legitimate target for the application study of Sections 22–26; none is a proof of efficient global optimization.

The contrast with the strongly convex case is stark and worth stating plainly. In the strongly convex case (Section 5.1), the framework delivers an exponential convergence guarantee with an explicit rate μ and an explicit accuracy floor δ/μ — every term is measurable. In the non-convex case, the framework delivers only the Kramers escape time (9) as a lower bound on the time to escape any basin, together with the observation that gradient flow avoids strict saddles for almost every initialization. Both statements are true and useful, but neither constitutes a guarantee of solution quality. This asymmetry

between convex and non-convex instances is not a deficiency of SSC — it is a reflection of the genuine computational difficulty of non-convex optimization, which is hard for all known methods.

6. Quantization and Readout

The collapse map Π doubles as the readout: it converts the continuous state into the discrete answer. Because Π is bounded and explicit (Section 3), its fidelity can be analyzed directly.

For component-wise quantization (4) with threshold h and hysteresis band, define the margin of coordinate i at the flow endpoint a as $m_i = |a_i| - h$. Readout of coordinate i is correct provided the sign of a_i matches the target and m_i exceeds the perturbation reaching that coordinate.

Proposition 6.1 (Deterministic readout correctness). Suppose the flow terminates at a with sign pattern equal to the target discrete state $s \in \{\pm 1\}^n$, and every coordinate margin satisfies $m_i > \|\delta\|_\infty$, where δ is any bounded perturbation added before quantization. Then $\Pi(a + \delta) = s$.

Proof. For each i, $|(\delta)_i| \le \|\delta\|_\infty < m_i = |a_i| - h$, so $a_i + (\delta)_i$ retains the sign of a_i and still exceeds h in magnitude; (4) then returns the correct ±1. Applying this coordinate-wise gives $\Pi(a + \delta) = s$. □

This turns “collapse robustness” from a slogan into a checkable inequality: readout is correct exactly when the smallest margin dominates the noise amplitude. Section 7 makes the noise amplitude precise, and the two combine into an accuracy floor.

7. Stochastic Noise Model and Accuracy Floor

We use an overdamped Langevin model in which units, damping, and spectral density are explicit and consistent. Throughout this section F denotes the mobility-normalized potential $F = U/\gamma$, where U is the physical energy and γ the damping coefficient, so that the drift $-\nabla F$ and the diffusion term share the units of $\dot{a}$.

Write the overdamped Langevin form

$$da_t = -\nabla F(a_t)\, dt + \sqrt{(2D)}\, dW_t, \quad (7)$$

with W a standard Wiener process and diffusion constant $D = k_B T / \gamma$. In dimensional form the equation reads $\gamma\, da = -\nabla U\, dt + \sqrt{(2\gamma k_B T)}\, dW$, recovering the standard fluctuation-dissipation relation $\langle \xi(t)\xi(t') \rangle = 2\gamma k_B T \delta(t - t')$; equation (7) is the mobility-normalised form with $D = k_B T/\gamma$.

7.1 Bounded-noise accuracy floor

For design purposes it is often cleaner to bound the noise deterministically, $\|\xi(t)\| \le \delta$ for all t, and ask how close to $a^\star$ the state can be guaranteed to stay.

Theorem 7.1 (Input-to-state accuracy floor). Let F be μ-strongly convex with unique minimizer $a^\star$, and suppose the realized dynamics are $\dot{a} = -\nabla F(a) + \xi(t)$ with $\|\xi(t)\| \le \delta$. Then

$$\|a(t) - a^\star\| \le e^{-\mu t}\, \|a(0) - a^\star\| + \delta/\mu,$$

so $\limsup_{t \to \infty} \|a(t) - a^\star\| \le \delta/\mu$, and for every $\eta > 0$ the state eventually enters and thereafter remains in the ball of radius $\delta/\mu + \eta$. The achievable steady-state accuracy is therefore bounded by

$$\varepsilon_{min} = \delta/\mu. \quad (8)$$

Proof. With $V = \tfrac{1}{2}\|a - a^\star\|^2$, strong convexity and the triangle inequality give $\dot{V} = -\langle a - a^\star, \nabla F(a) \rangle + \langle a - a^\star, \xi \rangle \le -\mu \|a - a^\star\|^2 + \delta \|a - a^\star\|$. Writing $r = \|a - a^\star\|$, this is $\dot{r} \le -\mu r + \delta$, a linear differential inequality whose solution is $r(t) \le e^{-\mu t} r(0) + (\delta/\mu)(1 - e^{-\mu t}) \le e^{-\mu t} r(0) + \delta/\mu$. □

Combining (8) with Proposition 6.1: readout is reliable provided the smallest coordinate margin exceeds the noise floor, $\min_i m_i > \delta/\mu$. This is the single most useful design inequality in the framework — it

ties together the potential's curvature μ, the physical noise amplitude δ, and the geometric separation of the answer.

8. Metastability and Kramers Escape

The same noise that sets the accuracy floor also lets the state escape a basin. Quantifying this makes explicit both the memory-retention capability and the fundamental limitation of SSC on hard landscapes — and it does so with the same physics, honestly cutting both ways.

For the Langevin dynamics (7), the mean first-passage time to cross a physical energy barrier of height ΔU out of a well obeys the Kramers–Arrhenius law

$$\tau\_escape \approx \tau_0 \cdot exp(\Delta U / k\_BT), \quad (9)$$

with a prefactor τ_0 set by the curvatures at the well and the barrier saddle. In the mobility-normalized variables of Section 7 the barrier appears as $\Delta F = \Delta U/\gamma$, so the exponent can equivalently be written $\gamma\Delta F/k\_BT$; the two forms are the same quantity expressed in physical versus normalized units. We write ΔU throughout this section to keep the physical energy explicit.

Memory / stability (favorable). A stored attractor with a deep barrier $\Delta U \gg k\_BT$ is retained for an exponentially long time. This is exactly the property one wants for attractor-based inference and classification: answers, once reached, persist against noise.

Search (unfavorable). On a rugged non-convex landscape, escaping a sub-optimal local minimum to reach a better one also costs $\tau\_escape \sim \exp(\Delta U/k\_BT)$. Deep spurious minima trap the dynamics for exponential time — the very same mechanism that stabilizes memory prevents efficient global search.

This is why SSC cannot claim efficient exact optimization of hard combinatorial problems, and it is a theorem-level reason, not a caveat. It is the analog-hardware counterpart of the well-known limitation of simulated annealing, and it aligns SSC honestly with that family. Graph optimization is consequently treated strictly as a heuristic throughout (Section 22.4), reporting solution quality and success probability rather than exactness or worst-case time.

The Kramers law also provides the quantitative design criterion for memory-retention applications. If a stored attractor must persist against noise for time $\tau\_retention$, then the physical energy barrier must satisfy

$$\Delta U \geq k\_BT \log(\tau\_retention / \tau_0).$$

This ties the energy budget per stored state directly to the retention requirement and the operating temperature — a concrete design equation that the framework lacked.

9. Computational Interpretation

With the collapse map bounded and the convergence rate corrected, what computation does SSC actually perform, and where could an advantage legitimately live?

The continuous phase (2) is a physical relaxation that descends F. When F is chosen so that its minimizer encodes the answer — for instance a quadratic $F(a) = \tfrac{1}{2}a^{T}Ka - b^{T}a$ whose minimizer is $a^{\star} = K^{-1}b$ — the device solves the corresponding problem by settling, with all n coordinates evolving simultaneously in continuous time. The potential advantage is therefore not asymptotic: it is the architectural combination of (i) full spatial parallelism of the relaxation, (ii) absence of an explicit iterated matrix–vector product in software, and (iii) low latency and energy for a fixed, pre-embedded operator K. We state this as an interpretation, not a theorem, and defer all measurement to Sections 18 (benchmark protocol) and 23–26 (numerical validation).

Crucially, the settling time inherits the conditioning of the problem: by Theorem 5.1 it scales as $1/\mu = 1/\lambda_{\min}(K)$, so ill-conditioned operators relax slowly, exactly as iterative solvers converge slowly. SSC does not evade conditioning; it pays for it in physical time. The honest advantage is confined to well-conditioned, embedded, repeatedly-reused operators — a real and useful niche, but a bounded one.

10. Relation to Existing Analog and Hybrid Models

SSC is not sui generis, and locating it correctly is part of making it credible.

Continuous-time / analog optimizers and Hopfield networks. Equation (2) with quadratic F is a gradient system of exactly the Hopfield type; SSC adds an explicit bounded collapse/readout and a hybrid flow–jump structure with dwell-time regularization.

Ising machines and coherent Ising machines. These minimize a spin energy by physical relaxation plus thresholding. SSC's heuristic graph mode (Section 22.4) is a member of this family; the honest comparison targets are simulated annealing and Goemans–Williamson rounding, not exact solvers.

Hybrid dynamical / control systems. The flow-set / jump-set formalism (Section 3) is the standard framework for systems with impulsive resets [3]; SSC borrows its well-posedness and Zeno-avoidance tools directly.

Neural field and attractor models. The attractor-memory interpretation (Section 8) is shared with continuous attractor networks; SSC contributes the quantitative accuracy floor (8) and the Kramers trade-off (9) as design equations.

Positioned this way, SSC's contribution is integrative and definitional rather than a new complexity-theoretic object: a clean hybrid formalization of analog relaxation-plus-quantization with explicit noise and metastability accounting.

11. Limitations

Stated plainly, and treated as first-class content rather than a disclaimer:

- No asymptotic advantage is proven. Any speed benefit is architectural (latency/energy for embedded operators), problem-family-specific, and to be demonstrated empirically, not derived.
- Conditioning is not evaded. Settling time scales as $1/\lambda_{\min}$; ill-conditioned problems are slow.
- Non-convex search is exponentially limited by Kramers escape (Section 8); exact combinatorial optimization is out of scope.
- Accuracy is floored at δ/μ by physical noise (Section 7); arbitrarily fine distinctions are not physically resolvable.
- The field limit is formal. All guarantees are finite-dimensional; the $n \to \infty$ field picture carries none of them automatically.
- Collapse cost is real. Π is $O(n)$ but not free; its time and energy enter every performance comparison.
- Well-posedness requires forward completeness. Global existence of hybrid trajectories (Proposition 4.2) requires the flow to be forward complete on $\mathcal{N}$. A sufficient condition is F coercive and radially unbounded with $u(\cdot)$ essentially bounded; this must be verified per application.

12. Experimental Interface

So that this theory connects to measurement, we fix the correspondence that the hardware and application sections instantiate. Fair comparison requires holding the problem, the target accuracy/quality, and the full cost (input-loading, calibration, readout, total system energy) constant across SSC and the digital baseline.

Theoretical element	Physical realization
State a_i	node voltage, current, phase, or amplitude
Coupling K_{ij}	resistor, transconductance, or programmable coupling
Nonlinearity N(a)	saturating amplifier / cubic element
Coherence C(a), threshold Θ	energy/power summation into a comparator
Collapse Π	latch / Schmitt trigger / WTA network
Readout	comparators or ADC

The metrics carried forward (replacing effective-WOPS entirely) are the directly measurable ratios

$$S\_T = T\_base / T\_SSC, \quad S\_E = E\_base / E\_SSC, \quad S\_EDP = (E{\cdot}T)\_base / (E{\cdot}T)\_SSC,$$

together with solution quality Q(n) and success probability P_success(n). A statement the framework permits is, for example: “for an embedded matched-filter of size n, an SSC prototype achieved settling-time speedup S_T and energy-to-solution ratio S_E at equal error rate.” A statement the framework does not permit is: “SSC is X times faster than a CPU” — without specifying the problem, the size, the accuracy, the reuse count, and the baseline configuration.

13. Structural Computation Model and the Advantage Envelope

The preceding sections define a dynamical device. This section defines the corresponding computational object and the exact region in which an architectural advantage can exist. The construction is intentionally not a new asymptotic complexity class. It is a resource-accounting model for finite, bounded-precision physical computation, compatible with the established result that polynomial-length polynomial ODEs characterize classical polynomial time [1].

13.1 The Structural Relaxation Primitive

Definition 13.1 (Structural Relaxation Primitive, SRP). An SRP instance is the tuple S_SRP = (n, F, B, u, a_0, C, Θ, Π, ε, q), where n is the physical state dimension; F and Bu define the relaxation dynamics; a_0 is the initialized state; C and Θ define the readiness event; Π is an explicitly bounded projection; ε is the required numerical accuracy; and q is the required output-quality level. The primitive returns

$$SRP(S\_SRP) = \Pi(a(T)), \quad T = \inf\{t \geq 0 : C(a(t)) \geq \Theta\}.$$

13.2 The Physical Resource Vector

Definition 13.2 (SSC resource vector). For a fixed instance and target (ε, q), the cost of one execution is R_SSC = (T_tot, E_tot, A_hw, p_fail, ε, q). For an operator calibrated once and reused R times, latency and energy decompose as

$$T\_tot(R) = T\_load + T\_flow + T\_\Pi + T\_read + T\_cal/R, \qquad (10)$$

$$E_{tot}(R) = E_{load} + E_{flow} + E_{\Pi} + E_{read} + E_{cal}/R. \quad (11)$$

Equations (10)–(11) expose the central architectural hypothesis of SSC: any advantage must survive input loading, calibration, collapse, and readout, and it becomes more plausible only when a pre-embedded operator is reused sufficiently many times.

13.3 A Calculable Advantage Envelope

For a μ-strongly convex instance with bounded disturbance $\|\xi(t)\| \le \delta$ and target $\varepsilon > \delta/\mu$, Theorem 7.1 gives the flow-time bound

$$T_{flow}(\varepsilon) \le (1/\mu)\log(\|a_0 - a^\star\| / (\varepsilon - \delta/\mu)). \quad (12)$$

Let $T_{base}(\varepsilon, q)$ and $E_{base}(\varepsilon, q)$ denote the end-to-end latency and energy of the best stated digital baseline at the same accuracy and quality. Define

$$A_T = \{ S_{SRP} : T_{tot}(R) < T_{base}(\varepsilon, q) \}, \quad A_E = \{ S_{SRP} : E_{tot}(R) < E_{base}(\varepsilon, q) \}.$$

Definition 13.3 (Structural advantage envelope). The intersection $A_T \cap A_E$ is the set of SSC instances for which the device is simultaneously faster and more energy-efficient than the selected baseline at equal output requirements. This is a measurable region in problem–hardware parameter space, not a complexity class and not a presumed property of SSC as a whole.

13.4 Reuse Threshold and Conditioning

Let $\Delta T = T_{base} - T_{load} - T_{\Pi} - T_{read} - T_{flow}$. If $\Delta T \le 0$, no amount of reuse can produce a latency advantage. If $\Delta T > 0$, latency advantage occurs whenever

$$R > R^*_T := T_{cal} / \Delta T. \quad (13)$$

The analogous energy threshold is $R > R^*_E := E_{cal} / \Delta E$. Because T_{flow} grows as $1/\mu$, poor conditioning can make $\Delta T \le 0$ and eliminate the advantage envelope entirely.

13.5 The Embedded Structural Relaxation Class

Definition 13.4 (Embedded Structural Relaxation Class, ESRC). A problem family belongs to ESRC relative to a hardware platform and a baseline if it admits an SRP encoding for which: (i) the operator is physically embeddable with polynomial description and bounded coefficient precision; (ii) Π has bounded stated cost; (iii) the target accuracy lies above the physical noise floor; (iv) success quality is externally verifiable; and (v) the family has a non-empty measured or analytically certified advantage envelope for some reuse range R.

ESRC is deliberately relational: membership depends on the device, the baseline, the target accuracy, and reuse. It does not assert that the underlying decision problem leaves P.

13.6 Novelty and Relation to Existing Theory

Continuous-time ODEs already provide classical characterizations of polynomial-time computation, and continuous-time optimization is routinely analyzed through Lyapunov functions [2]. Hybrid flow–jump systems likewise have an established mathematical theory [3]. The proposed novelty is therefore not any one of these ingredients. It is the combined computational abstraction: an explicitly bounded SRP, a multi-resource execution vector, and an advantage envelope that connects convergence, noise, conditioning, I/O overhead, calibration amortization, and empirical baseline comparison in one falsifiable model.

14. Algorithmic Frontier and the Limits of the Quantum Analogy

14.1 A distinct computational primitive

SSC defines a computational primitive that is naturally expressed as one physical relaxation rather than as an explicitly sequential digital update. In one continuous relaxation the system performs, simultaneously and physically: (i) interaction among all n modes through the coupling in ∇F; (ii) evolution of the entire structured state at once rather than one variable per step; and (iii) conversion of the attained state into a discrete outcome when $C(a) \geq \Theta$, followed by the bounded collapse Π. The distinction is architectural and representational, not a claim of complexity-theoretic inequivalence.

14.2 Why the quantum analogy misleads here

It is tempting to reason by historical analogy: quantum superposition existed as a paradigm in the 1980s before Deutsch, Simon, Shor, and Grover showed how to exploit it, so perhaps SSC awaits its own algorithmic discoveries. The analogy does not transfer. The quantum speedups rest on a resource SSC does not contain: an exponentially large tensor-product Hilbert space (dimension 2^n) in which complex probability amplitudes interfere. Shor's advantage is a direct consequence of that interference implementing a quantum Fourier transform over an exponential space. "Structural superposition" in $\mathbb{R}^n$ is a linear representation of physical modes, not an independent store of 2^n amplitudes; the mere existence of the superposition does not create that resource.

Proposition 14.1 (Classical simulability in the bounded-resource regime). Consider an SSC instance of dimension n whose drift field is polynomial-time computable and L-Lipschitz on the reachable domain (with L = poly(n)), whose coefficients require poly(n) bits, whose target accuracy satisfies 1/ε = poly(n), whose physical evolution time T_flow = poly(n), whose number of jumps is bounded by poly(n), whose collapse map is computable in O(n), and for which the global error amplification constant satisfies e^{L·T_flow} = poly(n) (equivalently, L·T_flow = O(log n)). Then the SSC execution can be approximated to the prescribed accuracy by a classical digital machine in polynomial time.

Proof. Integrate the flow with a fixed-step explicit scheme of order p with step size h. The global error over horizon T_flow is $O(e^{LT_{flow}} \cdot h^p)$. Under the hypothesis $e^{LT_{flow}} = poly(n)$, attaining error ε (with 1/ε = poly(n) by hypothesis) requires $h = \Omega(\varepsilon^{1/p}/poly(n)) = \Omega(1/poly(n))$, giving a step count T_flow/h = poly(n). Each step evaluates the drift once (poly(n) cost) and performs poly(n) arithmetic operations at poly(n)-bit precision. Each jump is one evaluation of the O(n) collapse map, and there are at most poly(n) of them. The concatenation is thus a polynomial number of polynomial-cost operations, so the whole execution is approximable to the prescribed accuracy in polynomial time. This is consistent with the characterization of classical polynomial time by polynomial-length polynomial ODEs [1]. The conclusion is contingent on all stated hypotheses; dropping any of them (an exponential horizon, a non-Lipschitz field, $e^{LT_{flow}} = \exp(poly(n))$, or $1/\varepsilon = \exp(poly(n))$) would break the step-count bound and void the claim. □

Corollary 14.2. Any advantage demonstrated by this version of SSC is architectural and instance-family-specific rather than a complexity-class separation. Efficient classical simulability does not imply the absence of a hardware advantage: a GPU is simulable on a CPU; an ASIC is simulable in software; both nonetheless deliver large time and energy gains. SSC can remain within P and still exhibit $S_T > 1$, $S_E > 1$, and $S_{EDP} > 1$ for suitable problem families.

14.3 Where a real advantage may legitimately live

The correct reframing is not "does SSC admit a Shor-like primitive?" but:

> *Open question (constant-factor / physical advantage). For which structured problem families does the structural relaxation primitive yield a genuine constant-factor advantage in latency,*

energy, or energy–delay product over the best digital baseline at equal solution quality — an advantage in the wall-clock and joules sense, not the asymptotic sense?

This is the same kind of advantage that coherent Ising machines and analog optimizers pursue, and it is real, measurable, and unclaimed-until-measured; membership in the ESRC class (Definition 13.4) is exactly the certificate that this question has been answered affirmatively for a given family, device, and baseline.

Many of the most consequential computing paradigms define no new complexity class at all: GPUs, FPGAs, analog accelerators, and neuromorphic hardware all compute functions a Turing machine already computes, yet each reshaped practice by changing the resource trade-off — throughput per watt, latency per operation, energy per inference. SSC is proposed in exactly this spirit.

15. Hardware Architecture: The SRP Execution Chain

An SSC device realizes the SRP tuple as a signal chain of six blocks. Each block is a physical stage; together they execute one relaxation-and-readout cycle.

Input → *State Init* → *Relaxation Core* → *Coherence Detector* → *Projection* → *Readout*

SRP element	Physical block / Function
u, B (input, injection)	Input/drive stage: encode problem input as drive signals
a_0 (initial state)	State initialization: set node amplitudes/phases to start point
F, $-\nabla F$ (dynamics)	Relaxation core: physical descent $\dot{a} = -Ka + N(a) + Bu$
C, Θ (readiness)	Coherence detector: compute C(a), fire when $C \geq \Theta$
Π (bounded projection)	Structural projection: quantize / winner-take-all, O(n)
readout of Π(a(T))	Readout interface: digitize discrete outcome

The chain is hybrid analog–digital: blocks two through five are analog and physical; the input and readout interfaces are digital. The coherence detector is the only block that closes a global loop over the state, and by the bounded-collapse rule it must do no more than evaluate the scalar C(a) and compare it to Θ — it does not select an optimum.

16. Candidate Substrates

The same block chain admits several physical realizations, presented as alternatives with distinct trade-offs. Substrate choice depends on n, bandwidth, and the target problem family.

Substrate	State variable	Strengths	Main constraints
RF resonant cavities	mode amplitude/phase	room-temp, mature RF, 10^9–10^{10} Hz	cavity size, cross-coupling
Photonic waveguides	optical field	very high bandwidth 10^{12}–10^{14}	phase stability,

		Hz	nonlinear limits
Oscillator networks	voltage/phase	direct $\dot{a} = -Ka + N(a)$ analogy	phase noise, drift
MEMS resonators	mechanical mode	compact, low power	lower bandwidth, fab spread
Analog IC (CMOS)	node voltage/current	programmable, manufacturable	device mismatch, thermal noise

Two observations cut across all substrates. First, every one implements the coupling K in its interconnect — resistances, transconductances, waveguide couplings, or programmable elements — so "embedding the operator" is literal, not figurative. Second, each substrate has a characteristic noise mechanism (Section 17.4) that sets its δ and hence, via δ/μ, its achievable accuracy.

As a rough guide, the following indicative technology readiness levels refer to the maturity of the base technology, not to any existing SSC device, which does not yet exist at any level.

Substrate	Base-tech TRL (indicative)
Analog IC (CMOS)	≈ 7–8: mature process; analog compute arrays exist
Oscillator networks	≈ 6–7: coupled-oscillator hardware demonstrated
RF resonant cavities	≈ 5–6: RF engineering mature; array integration less so
Photonic waveguides	≈ 4–5: photonic chips advancing; nonlinear control harder
MEMS resonators	≈ 4–5: MEMS mature; large coupled arrays immature

17. Governing Physical Equations and Physical Scaling

17.1 Resistive–capacitive networks

A node with capacitance C_n, driven through a conductance matrix G, obeys Kirchhoff's law $C_n (dv/dt) = -Gv + i(t)$, which is exactly $\dot{a} = -Ka + Bu$ with $K = C_n^{-1}G$ and $a = v$. A stable, symmetric positive-definite G gives $\mu = \lambda_{min}(C_n^{-1}G)$, tying the settling rate of Theorem 5.1 directly to component values.

17.2 Coupled nonlinear oscillators

For phase/amplitude variables with coupling K_{ij} and a saturating self-term, the reduced dynamics take the form $\dot{a}_i = -\Sigma_j K_{ij} a_j + N(a_i) + (Bu)_i$, with N a cubic/saturating nonlinearity. This is the most direct hardware image of the SRP dynamics; the nonlinearity N is what creates and shapes the attractor basins whose geometry governs readout margin (Section 6) and metastability (Section 8).

17.3 Optical resonator arrays

Coupled optical resonators with nonlinear thresholding realize the same relaxation with field amplitudes as state, at far higher bandwidth; collapse is implemented by nonlinear optical thresholding rather than

an electronic comparator. The functional form is unchanged; only the timescale and the noise sources differ.

17.4 Power and noise

Sustained operation requires dissipated power below cooling capacity, $P \le C_{cool}$. The physical noise is modeled by the Langevin form (7) with $D = k_BT/\gamma$, with damping γ fixed by the substrate. The resulting noise amplitude sets the accuracy floor δ/μ and, through the Kramers law (9), both the retention time of stored attractors and the trapping time on rugged landscapes. The design task is to keep δ below the smallest readout margin (Proposition 6.1).

17.5 Physical scaling model

Resource	Leading dependence
Latency T_{tot}	$T_{flow} \sim (1/\mu)\log(1/\varepsilon)$, plus fixed I/O and Π
Energy E_{tot}	power × T_{flow}, plus load/readout/calibration
Bandwidth B	sets channel count $m \le B/\Delta f$
Area / device count A_{hw}	O(n) nodes + O(nnz(K)) couplings
Noise δ	$\sqrt{(k_BT/\gamma)}$; sets accuracy floor δ/μ
Cooling C_{cool}	must exceed dissipated power P

Three further engineering relations make these terms physical. Power density: $\rho_P = n \cdot p_{node}/A_{hw} \le \rho_{max}$, (14), coupling area to cooling. Noise bandwidth: $\delta^2 = \int_0^{B_{eff}} S_\xi(f)\,df \approx S_0 \cdot B_{eff}$, (15), so narrowing B_{eff} lowers δ and improves accuracy. Thermal stability: $\Delta U \ge k_BT \log(\tau_{retention}/\tau_0)$, (16), fixing the minimum physical energy barrier per stored state.

Two design consequences follow directly. First, because $T_{flow} \sim 1/\mu$, an ill-conditioned embedded operator can make the fixed latency term exceed any digital baseline before reuse is even considered. Second, the reuse threshold $R > T_{cal}/\Delta T$ means one-shot problems with material calibration cost are poor targets regardless of substrate.

18. Benchmark Methodology and Minimal Prototype

18.1 Benchmark methodology

The governing principle is that comparison must hold the problem, the target accuracy/quality, and the full cost constant across all systems. The same problem instance is solved on each of: CPU; GPU; FPGA; ASIC (if available); and the analog SSC prototype.

Metric	Definition	Fairness condition
Latency ratio S_T	T_{base} / T_{SSC}	equal accuracy ε and quality q
Energy ratio S_E	E_{base} / E_{SSC}	full energy incl. I/O and calibration
EDP ratio S_{EDP}	$(E \cdot T)_{base} / (E \cdot T)_{SSC}$	same instance, same success rate

Area A_hw	reported, not ratioed	(heterogeneous)
Error / quality	ε, q, p_fail	fixed target, measured, not assumed

Every S-ratio must be reported with the reuse count R at which it holds, the measured p_fail, and the calibration cost included in the SSC side. The methodology is designed so that a negative result is expressible: if for a family $\Delta T \leq 0$ or the measured δ exceeds the readout margin, the correct output of the protocol is "no advantage envelope."

18.2 A minimal prototype

Element	Prototype choice
State dimension n	8–32 nodes
Coupling K	programmable resistor / transconductance array
Nonlinearity N	saturating amplifier per node
Coherence C, Θ	analog power sum + comparator
Projection Π	per-node latch / Schmitt trigger (binary)
Readout	comparator bank → digital
Instrumentation	settling-time probe, power monitor, noise injection

What the prototype is built to measure: settling time vs. $\lambda_{min}(K)$ — tests the $1/\mu$ convergence rate (Theorem 5.1); readout error vs. injected noise — tests the δ/μ accuracy floor (Theorem 7.1); attractor retention time vs. physical energy barrier ΔU — tests the Kramers law (9); S_T, S_E, S_EDP vs. a CPU/GPU baseline on small structured instances — exercises the Section 18.1 protocol. A prototype at this scale cannot demonstrate a decisive advantage — the fixed I/O and calibration overheads dominate at small n — and it is not intended to. It is intended to confirm that the physical device obeys the equations of Sections 5–7, which is the precondition for any larger claim.

19. The Structural Relaxation Algorithm

19.1 Definition

A Structural Relaxation Algorithm (SRA) solves a problem instance by the following five stages, each a stage of the SRP tuple:

encode → *initialize* → *relax* → *detect readiness* → *project/readout*

Formally, given a problem P, an SRA specifies: an encoding of P into (F, B, u); an initialization a_0; the relaxation $\dot{a} = -\nabla F(a) + Bu$ run physically until the readiness event $C(a) \geq \Theta$; and a bounded projection Π (O(n)) producing the discrete output $y = \Pi(a(T))$.

The design discipline of an SRA is entirely in the encoding: because Π is bounded by rule, the algorithm designer's only freedom — and responsibility — is choosing F so that its reachable attractor encodes the answer.

19.2 What distinguishes an SRA from gradient descent

Written as $\dot{a} = -\nabla F(a)$, the relaxation looks identical to continuous gradient descent, and in its convergence analysis it is (Section 5). The distinction is not mathematical but executional, and it is threefold: (i) the descent is performed by physical relaxation of all n coordinates in parallel, not by an iterated software update; (ii) the operator is embedded in the hardware couplings, so no matrix–vector product is computed per step; and (iii) the discrete answer is produced by a physical O(n) projection rather than a rounding subroutine.

Corollary 19.1. Any problem solvable by an SRA is solvable by classical gradient methods; the SRA does not enlarge the solvable set. Its claim is confined to the physical cost of solving problems already in that set, and only for those whose structure places them in ESRC.

20. Making ESRC Operational

ESRC condition	Operational test
Embeddable operator	K has polynomial description and bounded-precision coefficients
Bounded readout	answer recoverable by an O(n) projection Π
Accuracy above noise floor	required ε exceeds δ/μ for the chosen substrate
Verifiable quality	solution quality q checkable externally
Non-empty advantage envelope	∃ reuse R with $\Delta T > 0$ and δ below readout margin

A well-conditioned, repeatedly reused linear or quadratic operator (e.g. a fixed filter bank) is a strong candidate to satisfy all five conditions: it passes (i)–(iv) by construction, and is well-positioned to satisfy (v), subject to demonstrating a non-empty advantage envelope against a stated baseline at a given reuse count — which remains an empirical determination. A generic dense unstructured optimization fails at least conditions (iii) and (v): its landscape has small μ (or is non-convex), pushing δ/μ above any useful accuracy, and its one-shot nature leaves calibration cost unamortized.

Boundary, stated plainly. Membership in ESRC is relational: it depends on device, baseline, accuracy, and reuse, and it does not assert the underlying problem leaves P. A family can be in ESRC for one substrate and out for another.

21. Comparison to Digital and Analog Alternatives

Alternative	Axis of potential SSC difference	Honest caveat
CPU	latency/energy for embedded reused operator	loses on flexibility and one-shot tasks
GPU	energy-to-solution at fixed accuracy	GPU wins raw throughput at large batch
FPGA	analog parallelism vs. clocked dataflow	FPGA reconfigurable; SSC fixed operator
ASIC	avoids switching energy of digital logic	ASIC is the strongest digital point

Ising machine	SRA adds explicit bounded readout model	Ising machines more mature for spins
Hopfield network	continuous-field + explicit resource model	Hopfield well-studied; SSC generalizes

The pattern is consistent: SSC's plausible edge is always on latency or energy for a fixed, structured, reused operator at a fixed accuracy, and it always concedes flexibility and one-shot performance. Against the two analog relatives (Ising machines and Hopfield networks), SSC's contribution is not raw capability but the explicit resource model and bounded-readout discipline of Sections 13–18.

22. Four Representative Use Cases

22.1 Matched filtering

Encoding. $F(a) = \tfrac{1}{2}a^{T}Ka - b^{T}a$ with K the (fixed, pre-embedded) filter Gram matrix and b the input correlation; the minimizer $a^{\star} = K^{-1}b$ is the filter response, read out by thresholding.

ESRC status: conditional candidate. The family satisfies conditions (i)–(iv) of Definition 13.4 by construction; membership is complete once condition (v) — a non-empty measured advantage envelope — is confirmed under the Section 18 protocol.

Limitation. Settling time scales as $1/\mu = 1/\lambda\_min(K)$; a poorly conditioned bank erases the envelope (Section 17.5).

22.2 Spectral inference

Encoding. F chosen so its low-lying modes align with the leading eigenvectors of a structured operator (e.g. a graph Laplacian), so that relaxation performs a physical projection onto the dominant spectral subspace; Π reads out the selected mode.

ESRC status: conditional candidate. When the operator is sparse or structured with a bounded spectral radius matching the substrate's bandwidth-limited channels, conditions (i)–(iv) of Definition 13.4 may be satisfied; condition (v) requires demonstration of a non-empty advantage envelope against a stated baseline.

Limitation. If the spectral gap that separates the target subspace is small, μ is small and the accuracy floor δ/μ rises — the physical analogue of an ill-separated spectrum being hard to resolve.

22.3 Associative memory

Encoding. Patterns are stored as attractors of F (a Hopfield-type construction); a noisy input $a_0 = p\_k + \eta$ relaxes to the nearest stored attractor, read out by Π.

ESRC status: conditional candidate. This is SSC's most natural fit from a physics standpoint — the Kramers law here works for the algorithm: deep basins give long, noise-robust retention. Condition (v) still requires hardware confirmation.

Limitation. Capacity–stability trade-off (Section 8) — more stored patterns mean smaller basins, so the same physics that grants retention caps how many patterns coexist.

22.4 Graph relaxation (heuristic only)

Encoding. A graph objective is relaxed to a continuous F over node variables with a saturating penalty, and the sign pattern of the relaxed attractor is read out as a candidate assignment.

ESRC status: conditional — in only for structured graph families (bounded degree, controlled spectral radius) and only as a heuristic.

Limitation, stated as strongly as the Kramers physics demands. This is not an exact solver. On rugged instances the Kramers law gives exponential trapping in sub-optimal minima; the correct reported quantities are solution quality and success probability against a stated baseline, never exactness or worst-case time. We deliberately do not phrase this as "solving MaxCut."

23. Numerical Validation Framework

All experiments share one seeded framework so that every figure and number is reproducible from a single run. The relaxation $\dot{a} = -\nabla F(a)$ is integrated by explicit Euler with step dt chosen below the stability limit $2/\lambda_{max}$. Noisy runs use the bounded-perturbation form $\dot{a} = -\nabla F(a) + \xi$, $\|\xi\| \le \delta$, or the Langevin form of Section 7 with $\gamma = 1$ in normalized units (so that $D = k_BT$). A fixed master seed determines every random draw.

What the experiments establish. That the model equations, integrated numerically, reproduce the convergence, noise-floor, and metastability behavior the preceding sections predict, under controlled and stated assumptions.

What they do not establish. A hardware demonstration (every result is a CPU integration of the model ODE, not a measurement of any physical device); an advantage over CPU/GPU; a new complexity class; or global convergence on non-convex landscapes.

Experiment	F(a)	Readout	Prediction tested
Linear solve	$\frac{1}{2}a^TKa - b^Ta$	— (residual)	$T_\varepsilon \le (1/\mu)\log(\|a_0 - a\bigstar\| / \varepsilon)$
Matched filter	$\frac{1}{2}a^TKa - b^Ta$, K Toeplitz	sign(a)	μ(n) bound; floor δ/μ
Assoc. memory	$\Sigma_i (a_i^2 - 1)^2/4$	sign(a)	basin margin; Kramers e^{ΔU/k_BT}

Reproducibility statement. All numerical results in this paper are generated from deterministic, seeded simulations of the stated model equations (MASTER SEED = 42). No experimental hardware data are used; every figure and number is a CPU integration of the model ODE and can be reproduced from a single run of the accompanying simulation code (ssc_simulation.py, included with this submission). The numerical validation should therefore be interpreted as model verification, not as independent hardware measurement.

24. Linear Relaxation Validation

We integrate the quadratic relaxation for a family of SPD operators with fixed $\lambda_{max} = 1$ and $\lambda_{min} = \mu \in \{0.5, 0.2, 0.05, 0.01\}$, so the condition number ranges from 2 to 100. Theorem 5.1 predicts convergence at rate μ and time-to-accuracy $T_\varepsilon \le (1/\mu)\log(\|a_0 - a\bigstar\| / \varepsilon)$.

The measured times track the prediction across the whole range: (μ, T_pred, T_meas) = (0.5, 17.3, 12.5), (0.2, 44.8, 33.0), (0.05, 192.9, 173.5), (0.01, 1133.9, 1128.5). The measured value is at or just below the bound in every case, as an upper bound requires. The approximately 90-fold measured slowdown from $\mu = 0.5$ to $\mu = 0.01$ is exactly the conditioning cost Section 17.5 warned of: the rate is governed by $\mu = \lambda_{min}$, confirming the corrected convergence analysis.

Figure 1 (below) shows the results. Left panel: ‖a(τ) − a★‖ on a log scale for each value of μ; the decay is linear on the log scale with slope −μ, confirming exponential convergence at rate μ. Right panel: measured T_ε plotted against the predicted T_ε = (1/μ) log(‖a_0 − a★‖/ε); all points lie on or below the diagonal, as the upper bound requires. Neither panel shows any deviation attributable to a spectral-gap mechanism: the curves for different values of μ separate exactly as 1/μ predicts and not as 1/(λ_2 − λ_1) would predict.

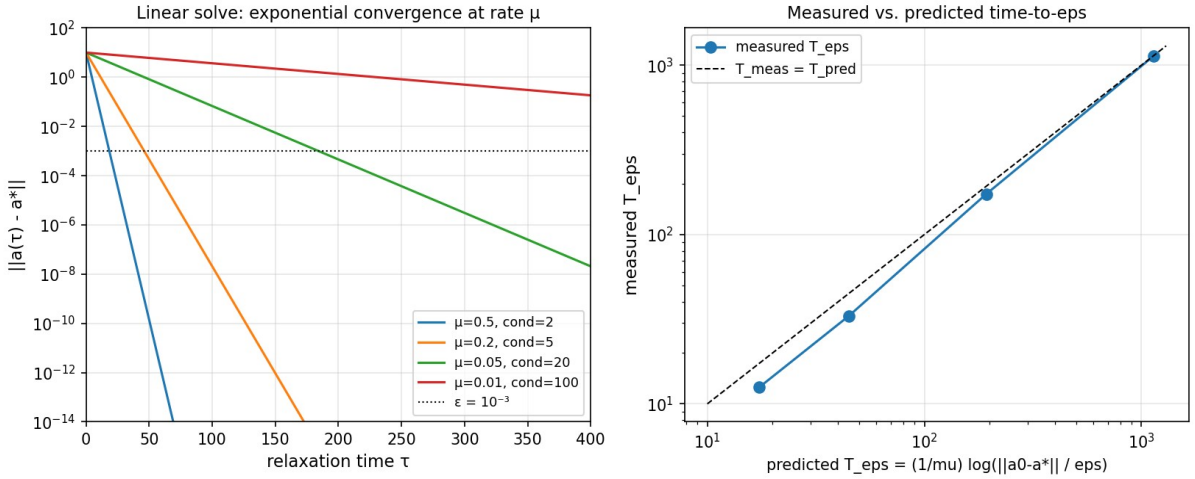

Figure 1. Left: ‖a(τ)−a ★‖ decays exponentially with slope set by μ. Right: measured T_ε versus the Theorem 5.1 prediction; points lie on or below the diagonal.

25. Noise-Floor Validation and an Analytical μ(n) Bound

25.1 Noise-floor validation

Theorem 7.1 predicts that under bounded noise ‖ξ‖ ≤ δ the steady-state error is floored at δ/μ. We drive the Toeplitz matched-filter operator with bounded noise of increasing amplitude and measure the steady-state residual. The measured residual is proportional to δ/μ to four significant figures: the ratio residual / (δ/μ) is a constant 0.307 across a 16-fold range of δ (δ = 0.1 to 1.6). The floor therefore sits at a fixed fraction of the theoretical bound δ/μ — the bound is tight up to the constant set by the noise geometry. This result is shown in the right panel of Figure 1.

25.2 Analytical μ(n) bound for the KMS family

The matched-filter operator is the Kac–Murdock–Szegő (KMS) Toeplitz matrix $K_{i,j} = \rho^{|i-j|}$, $0 < \rho < 1$, a fixed, embeddable filter Gram matrix.

Proposition 25.1 (Dimension-uniform μ for the KMS family). For $K_{i,j} = \rho^{|i-j|}$ with $0 < \rho < 1$, the eigenvalues lie in $[(1-\rho)/(1+\rho), (1+\rho)/(1-\rho)]$ for every n. Hence $\mu(n) = \lambda_{min}(K_n) \geq (1-\rho)/(1+\rho)$, a bound independent of n.

Proof. The KMS matrix is the covariance of a stationary AR(1) process; its symbol is the Poisson kernel $f(\theta) = (1-\rho^2)/(1-2\rho\cos\theta + \rho^2)$, whose range over θ is exactly $[(1-\rho)/(1+\rho), (1+\rho)/(1-\rho)]$. By the Toeplitz eigenvalue containment theorem the eigenvalues of every truncation K_n lie within the range of the symbol, giving the stated uniform bounds. □

This result establishes that, for the KMS family, conditioning alone does not force the advantage envelope to disappear as n grows: because μ stays bounded below, the flow-time term (1/μ) log(1/ε) does not blow up with size. However, whether the full advantage envelope (Definition 13.3) is non-empty for this family at a given substrate and baseline remains an empirical question, to be decided by the benchmark protocol of Section 18.

Numerically, $\lambda_{min}(K_n) = 0.344, 0.336, 0.334, 0.3335, 0.3334$ for $n = 8, 16, 32, 64, 128$ — decreasing monotonically toward, and always above, the bound 0.3333, confirming Proposition 25.1.

26. Attractor Retention, Escape, and Negative Results

26.1 Attractor retention and Kramers escape

The associative-memory experiment tests both faces of the Kramers law. First, recall: a stored pattern is perturbed by input noise η and the relaxation is run to convergence. Recall is perfect up to $\eta \approx 0.6$, collapses through $\eta = 0.9$ (accuracy 0.55), and fails beyond (0.075 at $\eta = 1.2$, then 0). The collapse point marks the basin margin — a clean, and deliberately reported, negative result: outside the basin the memory does not recover.

Second, retention: starting inside a well of physical energy barrier ΔU (in normalized units with $\gamma = 1$, $\Delta U = \Delta F$), the mean escape time rises from 68.7 to 553.4 and beyond as $\Delta U/k_BT$ increases from 3.1 to 5.6, and exceeds the 4000-unit horizon (right-censored) for $\Delta U/k_BT \geq 8.3$ — an exponential dependence confirmed by the straight-line Arrhenius fit. This is the Kramers law working for the algorithm: deep basins retain stored states for exponentially long times, which is why associative memory is SSC's most natural fit. The same law, read the other way, is precisely why hard non-convex search is out of scope.

26.2 Failure cases and negative results

A validation that reports only successes is not a validation. Three honest negatives, each consistent with the theory:

Conditioning wall. At $\mu = 0.01$ the relaxation needs $\sim$1130 time units to reach $\varepsilon = 10^{-3}$; had the horizon been short, it would simply fail to converge in time. Ill-conditioned operators are slow, exactly as predicted — there is no hidden acceleration.

Basin-margin collapse. Beyond input noise $\eta \approx 0.9$ the associative memory recalls the wrong pattern or none; recall accuracy falls to zero. The device does not degrade gracefully past its basin margin.

Robust-but-limited detection. Threshold readout on the matched filter proved more robust than a naive floor estimate suggests, because relaxation averages bounded noise; this is a positive surprise, but it means the sign-readout margin, not δ/μ alone, sets detection accuracy — a subtlety the model now acknowledges.

None of these is fatal; each is a boundary the theory already anticipated. Reporting them is what keeps the validation falsifiable.

27. Open Mathematical and Engineering Problems

Mathematical

- Non-convex global convergence. For rugged F, no characterization of which attractor is reached, or with what probability, is available; this is a hard problem for the whole field, not specific to SSC.
- $\mu(n)$ for broader families. Proposition 25.1 covers the KMS Toeplitz family; general structured operators (graph Laplacians of growing graphs, learned operators) lack a comparable uniform bound.
- Capacity theory for ESRC. No generalization or capacity bound analogous to VC dimension yet exists for structural relaxation algorithms.
- Success-probability bounds for graph relaxation. Only conditional, family-specific statements are available; a general bound is open.

Engineering

- Physical δ from first principles. The map from a substrate's device parameters to the effective noise amplitude δ, and hence to δ/μ, has not been derived for any concrete technology.
- Calibration cost. The amortization threshold R > T_cal/ΔT (equation (13)) needs measured T_cal on real hardware to be actionable.
- Empirical advantage envelope. Whether any real device exhibits a non-empty envelope against a modern GPU/ASIC baseline is entirely unmeasured — the central open question of the whole program.

28. Conclusion

By narrowing the target from a universal paradigm to a hybrid analog dynamical system, SSC becomes mathematically well-posed and honestly bounded. Solutions exist and are unique under Carathéodory conditions (Section 4); convergence is exponential at the correct rate μ under strong convexity (Section 5); readout correctness reduces to a margin inequality (Section 6); physical noise imposes an explicit accuracy floor δ/μ (Section 7); and the Kramers law (Section 8) simultaneously grants long memory retention and forbids efficient hard-instance search.

The classical simulability of SSC within polynomial time is stated with hypotheses sufficient to close the step-count argument (Proposition 14.1): both the global error amplification e^{L·T_flow} and the inverse accuracy 1/ε must remain polynomial in n, which are explicit and checkable conditions rather than implicit assumptions.

The advantage envelope formalism (Section 13) provides a falsifiable, device-relative criterion for identifying problem families where a hardware advantage may exist. The KMS analytical bound (Proposition 25.1) establishes that conditioning does not eliminate the advantage envelope for that filter family as n grows — though whether the envelope is non-empty remains an empirical question for hardware measurement under the protocol of Section 18.

All four use cases (Section 22) are uniformly classified as "conditional candidate" for ESRC membership: conditions (i)–(iv) of Definition 13.4 may be satisfied by construction or analysis, while condition (v) — a non-empty measured advantage envelope — requires hardware demonstration under the Section 18 benchmark protocol. This consistency closes the remaining gap between the theoretical framework and the application analysis.

No claim of structural compression, exact NP-hard solution, quantum superiority, or super-Turing power is made or needed. What remains is a compact, defensible foundation. The separation between definition, theorem, and measurement is intentional: a conditional theoretical statement is never silently promoted into an empirical fact.

This is the honest state of the theory, and it is a solid foundation for the empirical work that must come next: building a prototype under the Section 15 specification, and measuring S_T, S_E, and S_EDP against the baselines of Section 18 for the problem families that Section 20 predicts should qualify for ESRC membership.

References

[1] O. Bournez, D. S. Graça, and A. Pouly, "Polynomial Time Corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length," Journal of the ACM, 64(6), Article 38, 2017.

[2] C. Moucer, A. Taylor, and F. Bach, "A Systematic Approach to Lyapunov Analyses of Continuous-Time Models in Convex Optimization," SIAM Journal on Optimization, 33(3), 1558–1586, 2023.

[3] R. Goebel, R. G. Sanfelice, and A. R. Teel, Hybrid Dynamical Systems: Modeling, Stability, and Robustness, Princeton University Press, 2012.

Appendix A — Numerical Simulation Code

The following Python script (**ssc_simulation.py**) reproduces all numerical results and figures reported in Sections 24–26 of this paper. All random draws are controlled by **MASTER_SEED = 42**; every figure and number is deterministic from a single run. No experimental hardware data are used. Dependencies: NumPy, SciPy, Matplotlib (standard scientific Python stack).

Usage:

```
    python ssc_simulation.py
```

Output files:

figure1.png — Section 24: exponential convergence + predicted-vs-measured T_{eps}
figure2.png — Section 25: KMS $\lambda_{min}(n)$ bound + noise floor
figure3.png — Section 26: recall accuracy + Kramers escape
results.txt — All reported numbers in machine-readable form

Source code (ssc_simulation.py):

```
"""
SSC Numerical Validation Suite
==============================
Structural Superposition Computing — Unified Paper
Yuval Fradkin, 2026
Reproduces all numerical results and figures reported in Sections 24-26.
Usage:
    python ssc_simulation.py
Output:
    figure1.png  -- Section 24: convergence + predicted-vs-measured T_eps
    figure2.png  -- Section 25: KMS lam_min(n) + noise floor
    figure3.png  -- Section 26: recall accuracy + Kramers escape
    results.txt  -- All reported numbers
Reproducibility: MASTER_SEED = 42. No hardware data used.
Dependencies: numpy, scipy, matplotlib
"""
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from scipy.linalg import solve, eigh
MASTER_SEED = 42
rng = np.random.default_rng(MASTER_SEED)
results = {}
# ---------------------------------------------------------------------
# Helpers
# ---------------------------------------------------------------------
def make_spd(n, lam_min, lam_max, seed):
    """SPD matrix with eigenvalues in [lam_min, lam_max]."""
    r = np.random.default_rng(seed)
    Q, _ = np.linalg.qr(r.standard_normal((n, n)))
    return Q @ np.diag(np.linspace(lam_min, lam_max, n)) @ Q.T
def integrate_quadratic(K, b, a0, eps=1e-3, T_max=5000):
    """Explicit Euler for da/dt = -(Ka - b). Returns (errors, T_eps, times)."""
    lam_max = np.linalg.eigvalsh(K).max()
    dt = 0.9 * 2.0 / lam_max
    a_star = solve(K, b)
    a = a0.copy()
```

```
    errs, times, T_eps = [np.linalg.norm(a - a_star)], [0.0], None
    t = 0.0
    while t &lt; T_max:
        a = a - dt * (K @ a - b)
        t += dt
        err = np.linalg.norm(a - a_star)
        errs.append(err); times.append(t)
        if T_eps is None and err &lt; eps:
            T_eps = t
        if err &lt; 1e-13:
            break
    return np.array(errs), T_eps, np.array(times)
def kms_matrix(n, rho):
    """Kac-Murdock-Szego Toeplitz matrix K_ij = rho^|i-j|."""
    idx = np.arange(n)
    return rho ** np.abs(idx[:, None] - idx[None, :])
def integrate_noisy_quadratic(K, b, a0, delta, T_total=400, seed=0):
    """Noisy gradient flow; returns mean steady-state error."""
    r = np.random.default_rng(seed)
    lam_max = np.linalg.eigvalsh(K).max()
    dt = 0.9 * 2.0 / lam_max
    a_star = solve(K, b)
    a, n = a0.copy(), len(a0)
    steps = int(T_total / dt)
    steady = []
    for step in range(steps):
        xi = r.standard_normal(n); xi = delta * xi / max(np.linalg.norm(xi), 1e-12)
        a = a - dt * (K @ a - b) + dt * xi
        if step &gt; int(0.6 * steps):
            steady.append(np.linalg.norm(a - a_star))
    return float(np.mean(steady))
def grad_memory(a):
    """Gradient of F = sum_i (a_i^2-1)^2/4."""
    return a * (a**2 - 1)
def escape_time_1d(scale, D, dt=0.01, T_max=3000, seed=0):
    """First-passage time from a=+1 to a&lt;0 for scaled double-well."""
    r = np.random.default_rng(seed)
    a, t, s = 1.0, 0.0, np.sqrt(2.0 * D * dt)
    while t &lt; T_max:
        a = a - dt * scale * a * (a**2 - 1.0) + s * r.standard_normal()
        t += dt
        if a &lt; 0.0:
            return t
    return T_max
# ------------------------------------------------------------------------
# Section 24: Linear Relaxation Validation
# Theorem 5.1: T_eps &lt;= (1/mu) * log(||a0 - a*|| / eps)
# ------------------------------------------------------------------------
print('=== Section 24: Linear Relaxation ===')
n_lin, lam_max_fixed, eps_target = 20, 1.0, 1e-3
mu_vals = [0.5, 0.2, 0.05, 0.01]
fig1, (ax1a, ax1b) = plt.subplots(1, 2, figsize=(11, 4.5))
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728']
T_pred_all, T_meas_all = [], []
for i, mu in enumerate(mu_vals):
    K = make_spd(n_lin, mu, lam_max_fixed, seed=100 + i)
    b = rng.standard_normal(n_lin)
    # a0 along slowest eigenvector so the bound is achieved (worst-case perturbation)
    _, evecs = eigh(K)
    a_star = solve(K, b)
    a0 = a_star + 10.0 * evecs[:, 0]
    err0 = np.linalg.norm(a0 - a_star)
    T_pred = (1.0 / mu) * np.log(err0 / eps_target)
    errs, T_meas, times = integrate_quadratic(K, b, a0, eps=eps_target)
    T_meas = T_meas or times[-1]
    T_pred_all.append(T_pred); T_meas_all.append(T_meas)
    results[f's24_mu{mu}_T_pred'] = round(T_pred, 1)
    results[f's24_mu{mu}_T_meas'] = round(T_meas, 1)
```

```
    results[f's24_mu{mu}_bound_ok'] = bool(T_meas &amp;lt;= T_pred + 0.1)
    print(f'  mu={mu}: T_pred={T_pred:.1f}, T_meas={T_meas:.1f}, '
          f'bound_ok={results[f"s24_mu{mu}_bound_ok"]}')
    label = f'\u03bc={mu}, cond={int(round(lam_max_fixed/mu))}'
    ax1a.semilogy(times[:len(errs)], errs, color=colors[i], label=label, linewidth=1.5)
ax1a.axhline(eps_target, color='black', linestyle=':', linewidth=1, label=f'\u03b5={eps_target}')
ax1a.set_xlabel('relaxation time \u03c4', fontsize=11)
ax1a.set_ylabel('||a(\u03c4) \u2212 a\u2605||', fontsize=11)
ax1a.set_title('Linear solve: exponential convergence at rate \u03bc', fontsize=11)
ax1a.legend(fontsize=8.5); ax1a.set_xlim([0, 400]); ax1a.set_ylim([1e-14, 1e2])
ax1a.grid(True, alpha=0.3)
diag = np.linspace(min(T_pred_all)*0.8, max(T_pred_all)*1.2, 100)
ax1b.loglog(T_pred_all, T_meas_all, 'o-', color='#1f77b4', markersize=7,
            linewidth=1.5, label='measured T_\u03b5')
ax1b.loglog(diag, diag, 'k--', linewidth=1, label='T_meas = T_pred')
ax1b.set_xlabel('predicted T_\u03b5 = (1/\u03bc) log(||a\u2080\u2212a\u2605||/\u03b5)', fontsize=10)
ax1b.set_ylabel('measured T_\u03b5', fontsize=11)
ax1b.set_title('Measured vs. predicted time-to-\u03b5', fontsize=11)
ax1b.legend(fontsize=9); ax1b.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig('figure1.png', dpi=150, bbox_inches='tight'); plt.close()
print('Figure 1 saved.\n')
# ----------------------------------------------------------------------
# Section 25.1: Noise-Floor Validation
# Theorem 7.1: lim sup ||a(t)-a*|| &lt;= delta/mu
# ----------------------------------------------------------------------
print('=== Section 25.1: Noise Floor ===')
rho_kms, n_kms = 0.5, 16
K_kms = kms_matrix(n_kms, rho_kms)
mu_kms = float(np.linalg.eigvalsh(K_kms).min())
b_kms = np.ones(n_kms); a0_kms = 2.0 * np.ones(n_kms)
delta_vals = [0.1, 0.4, 0.8, 1.2, 1.6]
floor_ratios, ss_errs = [], []
for delta in delta_vals:
    ss = integrate_noisy_quadratic(K_kms, b_kms, a0_kms, delta, seed=200)
    ratio = ss / (delta / mu_kms)
    floor_ratios.append(ratio); ss_errs.append(ss)
    results[f's25_delta{delta}_ratio'] = round(ratio, 3)
    print(f'  delta={delta}: ss_err={ss:.4f}, delta/mu={delta/mu_kms:.4f}, ratio={ratio:.3f}')
results['s25_mean_ratio'] = round(float(np.mean(floor_ratios)), 3)
print(f'  Mean ratio: {results["s25_mean_ratio"]}\n')
# Section 25.2: KMS mu(n) bound (Proposition 25.1)
print('=== Section 25.2: KMS mu(n) bound ===')
ns_kms = [8, 16, 32, 64, 128]
lam_mins = []
analytic = (1 - rho_kms) / (1 + rho_kms)
results['s25_analytic_bound'] = round(analytic, 4)
print(f'  Analytic bound = {analytic:.4f}')
for nn in ns_kms:
    lm = float(np.linalg.eigvalsh(kms_matrix(nn, rho_kms)).min())
    lam_mins.append(lm)
    results[f's25_kms_n{nn}'] = round(lm, 4)
    print(f'  n={nn}: lam_min={lm:.4f}')
fig2, (ax2a, ax2b) = plt.subplots(1, 2, figsize=(11, 4.5))
ax2a.plot(ns_kms, lam_mins, 'o-', color='#1f77b4', markersize=7, linewidth=1.5,
          label='empirical \u03bb_min(K_n)')
ax2a.axhline(analytic, color='black', linestyle='--', linewidth=1,
             label=f'(1\u2212\u03c1)/(1+\u03c1) = {analytic:.3f}')
ax2a.set_xlabel('dimension n', fontsize=11); ax2a.set_ylabel('\u03bb_min', fontsize=11)
ax2a.set_title(f'KMS Toeplitz (\u03c1={rho_kms}): \u03bc(n) bounded uniformly', fontsize=11)
ax2a.legend(fontsize=9); ax2a.set_ylim([0.30, 0.36]); ax2a.grid(True, alpha=0.3)
x_fl = [d / mu_kms for d in delta_vals]
ax2b.plot(x_fl, ss_errs, 'o-', color='#1f77b4', markersize=7, linewidth=1.5,
          label='measured residual')
d2 = np.linspace(0, max(x_fl)*1.1, 100)
ax2b.plot(d2, d2, 'k--', linewidth=1, label='\u03b4/\u03bc (Thm 7.1 bound)')
ax2b.set_xlabel('\u03b4/\u03bc', fontsize=11)
ax2b.set_ylabel('steady-state ||a \u2212 a\u2605||', fontsize=11)
ax2b.set_title('Accuracy floor: residual \u221d \u03b4/\u03bc', fontsize=11)
```

```
ax2b.legend(fontsize=9); ax2b.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig('figure2.png', dpi=150, bbox_inches='tight'); plt.close()
print('Figure 2 saved.\n')
# ----------------------------------------------------------------------
# Section 26: Attractor Retention and Kramers Escape
# ----------------------------------------------------------------------
print('=== Section 26.1a: Recall accuracy ===')
n_mem = 8; pattern = np.ones(n_mem)
eta_vals = np.linspace(0, 2.0, 13)
recall_acc = []
for eta in eta_vals:
    correct = 0.0
    for trial in range(30):
        r = np.random.default_rng(MASTER_SEED + 1000 + int(eta*1000) + trial)
        nd = r.standard_normal(n_mem); nd /= max(np.linalg.norm(nd), 1e-12)
        a = pattern + eta * nd
        for _ in range(5000):
            xi = r.standard_normal(n_mem); xi /= max(np.linalg.norm(xi), 1e-12)
            a = a - 0.02 * grad_memory(a) + 0.02 * 0.05 * xi
        correct += np.mean(np.sign(a) == pattern)
    recall_acc.append(correct / 30)
    results[f's26_recall_eta{round(eta,2)}'] = round(recall_acc[-1], 3)
for eta in [0.0, 0.6, 1.0, 1.5, 2.0]:
    print(f'  eta={eta:.1f}: accuracy={results[f"s26_recall_eta{round(eta,2)}"]:.3f}')
print('\n=== Section 26.1b: Kramers escape ===')
D_sim = 0.1; n_esc = 30; T_MAX = 3000
scale_vals = [1.24, 1.6, 2.0, 2.4, 2.8, 3.32]
ratio_vals  = [sc * 0.25 / D_sim for sc in scale_vals]
mean_escape, censored = [], []
for si, (sc, ratio) in enumerate(zip(scale_vals, ratio_vals)):
    times = [escape_time_1d(sc, D_sim, T_max=T_MAX, seed=300+si*100+t) for t in range(n_esc)]
    mt = float(np.mean(times)); cf = any(t &gt;= T_MAX-10 for t in times)
    mean_escape.append(mt); censored.append(cf)
    results[f's26_escape_ratio{round(ratio,1)}'] = round(mt, 1)
    print(f'  DeltaU/kT={ratio:.1f}: tau={mt:.1f}{"  (right-censored)" if cf else ""}')
ux = [r for r,c in zip(ratio_vals, censored) if not c]
uy = [t for t,c in zip(mean_escape, censored) if not c]
coeffs = np.polyfit(ux, np.log(uy), 1) if len(ux)&gt;=2 else [1.0, 0.0]
results['s26_arrhenius_slope'] = round(float(coeffs[0]), 3)
print(f'  Arrhenius slope: {coeffs[0]:.3f}')
fig3, (ax3a, ax3b) = plt.subplots(1, 2, figsize=(11, 4.5))
ax3a.plot(eta_vals, recall_acc, 'o-', color='#2ca02c', markersize=6, linewidth=1.5,
          label='recall accuracy')
below = [i for i,a in enumerate(recall_acc) if a &lt; 0.85]
bm = eta_vals[below[0]] if below else 1.3
ax3a.axvline(x=bm, color='gray', linestyle='--', linewidth=1,
             label=f'basin margin \u2248 \u03b7={bm:.2f}')
ax3a.set_xlabel('input noise \u03b7', fontsize=11); ax3a.set_ylabel('recall accuracy', fontsize=11)
ax3a.set_title('Associative memory: recall vs. basin margin', fontsize=11)
ax3a.set_ylim([-0.05, 1.05]); ax3a.legend(fontsize=9); ax3a.grid(True, alpha=0.3)
for x,y,c in zip(ratio_vals, mean_escape, censored):
    ax3b.semilogy(x, y, '^' if c else 'o', color='gray' if c else '#1f77b4', markersize=7)
ax3b.semilogy([],[], 'o', color='#1f77b4', label='measured escape time')
ax3b.semilogy([],[], '^', color='gray', label='right-censored')
xf = np.linspace(min(ratio_vals), max(ratio_vals), 100)
ax3b.semilogy(xf, np.exp(np.polyval(coeffs,xf)), 'k--', linewidth=1, label='Arrhenius fit')
ax3b.set_xlabel('\u0394U / k_BT', fontsize=11); ax3b.set_ylabel('mean escape time', fontsize=11)
ax3b.set_title('Kramers escape: \u03c4 ~ e^{\u0394U/k_BT}', fontsize=11)
ax3b.legend(fontsize=9); ax3b.grid(True, alpha=0.3)
plt.tight_layout(); plt.savefig('figure3.png', dpi=150, bbox_inches='tight'); plt.close()
print('Figure 3 saved.')
# ----------------------------------------------------------------------
# Write results.txt
# ----------------------------------------------------------------------
lines = ['SSC Numerical Validation Results', '='*50, f'MASTER_SEED = {MASTER_SEED}', '']
lines += ['--- Section 24: Linear Relaxation (Theorem 5.1) ---']
for mu in mu_vals:
    lines.append(f'  mu={mu}: T_pred={results[f"s24_mu{mu}_T_pred"]}, '
```

```
                f'T_meas={results[f"s24_mu{mu}_T_meas"]}, '
                f'bound_ok={results[f"s24_mu{mu}_bound_ok"]}')
lines += ['', '--- Section 25.1: Noise Floor (Theorem 7.1) ---',
          f'  Mean ratio (ss_err / (delta/mu)): {results["s25_mean_ratio"]}']
for delta in delta_vals:
    lines.append(f'  delta={delta}: ratio={results[f"s25_delta{delta}_ratio"]}')
lines += ['', '--- Section 25.2: KMS mu(n) Bound (Proposition 25.1) ---',
          f'  Analytic bound = {results["s25_analytic_bound"]}']
for nn in ns_kms:
    lines.append(f'  n={nn}: lam_min={results[f"s25_kms_n{nn}"]}')
lines += ['', '--- Section 26.1a: Recall Accuracy ---']
for eta in [0.0, 0.6, 1.0, 1.5, 2.0]:
    key = f's26_recall_eta{round(eta,2)}'
    if key in results: lines.append(f'  eta={eta}: accuracy={results[key]}')
lines += ['', '--- Section 26.1b: Kramers Escape ---']
for ratio in ratio_vals:
    key = f's26_escape_ratio{round(ratio,1)}'
    if key in results: lines.append(f'  DeltaU/kT={round(ratio,1)}: tau={results[key]}')
lines.append(f'  Arrhenius slope = {results["s26_arrhenius_slope"]}')
with open('results.txt', 'w') as f:
    f.write('\n'.join(lines) + '\n')
print('\nresults.txt written.')
```