

Supplementary Material for: “Worldline Susceptibility as a Surrogate for Spectral Gaps in Quantum Annealing Schedule Design”

The `qanneal` Library: Theoretical Foundations, Algorithms, and Software

Suraj Singh,^{1,2,3,*} Lakshya Nagpal,^{1,2,3,*} Vikas Chauhan,^{4,2} and S. R. Hassan^{1,2,5}

¹The Institute of Mathematical Sciences (IMSc), C.I.T Campus, Taramani, Chennai 600113, India

²QCAR Group, The Institute of Mathematical Sciences (IMSc), Chennai 600113, India

³Pecslab Research

⁴Department of Physics, Ramjas College, University of Delhi, Delhi 110007, India

⁵Homi Bhabha National Institute, Anushakti Nagar, Mumbai, Maharashtra 400094

*These authors contributed equally to this work.

Abstract

Part of the numerical pipeline behind the main text — the classical Monte Carlo measurement of the worldline magnetization susceptibility χ_m and the deterministic construction that turns it into an annealing schedule (main-text Eqs. 7–15) — was carried out with a worldline-QMC implementation developed by the authors. This supplement documents that implementation in full, alongside the statistical-mechanical foundations it rests on. The *other* part of the main text’s pipeline — exact diagonalization of the spectral gap $\Delta(s)$, the exact Roland–Cerf schedule built from it, and the unitary/Lindblad time evolution used to compute ground-state success probabilities — was produced with separate analysis code that is not part of the software described here; Sec. S11 states its numerical protocol for completeness but is not a description of this library. Since the initial submission, the worldline-susceptibility method has also been generalized into `qanneal`, an open-source Ising/QUBO annealing library, as a standalone, production-quality solver (`method="sqa_chi"`) applicable to arbitrary dense or sparse problems, not only the Sherrington–Kirkpatrick ensemble studied here; Sec. S10 states precisely how this released version relates to the analysis that produced the main text’s figures. This supplement is organized the way the library is organized: from the statistical mechanics every engine rests on, through the five annealing engines it provides, to the two susceptibility-paced adaptive schedules it implements, one of which generalizes the method of the main text. A concordance table (Sec. S13) summarizes, item by item, which main-text results this software did and did not produce, and a condensed API reference and symbol glossary (Secs. S14–S15) close the supplement. `qanneal` is released under the Apache License 2.0.

Contents

S1 The Library as This Work’s Instrument	2
S2 Mathematical Foundations: Ising and QUBO	3
S3 Statistical Mechanics and the Metropolis Rule	4
S4 Engine I — Simulated Annealing (<code>sa</code>)	4

S5 Engine II — Simulated Quantum Annealing (<code>sqa</code>) and the Suzuki–Trotter Mapping	5
S5.1 From the transverse-field Ising model to a classical action	5
S5.2 Monte Carlo moves and why more than one move type is needed	5
S6 Engine III — Quantum Parallel Tempering (<code>sqapt</code>)	6
S7 Engine IV — Continuous-Time Path-Integral Monte Carlo (<code>ctpimc</code>)	6
S8 Adaptive Schedules I: The Bond-Susceptibility Optimal $J_{\perp}$ Schedule	7
S8.1 From the gap to a measurable quantity	7
S8.2 Budget calibration and the inverse-CDF trajectory	7
S8.3 Safeguards	8
S9 Adaptive Schedules II: The Worldline-Susceptibility Schedule (<code>sqa_chi</code>)	8
S9.1 The pilot scan	8
S9.2 From the scan to a schedule: floor regularization and inverse-CDF placement	8
S9.3 The parity-parallel checkerboard kernel	9
S9.4 Two-phase protocol and production anneal	9
S9.5 Relationship between the two adaptive schedules	10
S10 Relationship Between This Release and the Main Text’s Original Analysis	11
S11 Numerical Protocol for the Reference Roland–Cerf Schedule	11
S12 Parallel and HPC Deployment	12
S13 Concordance: Main-Text Results and the Methods Behind Them	12
S14 Condensed API Reference	12
S15 Symbol Glossary	13
S16 Code and Data Availability	13

S1 The Library as This Work’s Instrument

The central empirical claim of the main text — that a computationally inexpensive worldline observable, measured during a classical Monte Carlo simulation, can substitute for the exponentially expensive spectral gap in annealing schedule design, and can even outperform the theoretically optimal Roland–Cerf schedule built from that gap — rests on two logically separate numerical pipelines. *Only the first is documented in this supplement.*

1. **Measuring χ_m and constructing the surrogate schedule** (main-text Eqs. 7–15): a worldline-QMC simulation of the transverse-field Ising model, at a fixed inverse temperature, used purely to measure $\chi_m(s)$ and turn it into a schedule. This is what the software described in this supplement does.
2. **Grading a schedule** once constructed: exact diagonalization of $\Delta(s)$, the resulting exact Roland–Cerf schedule (main-text Eq. 20), and unitary/Lindblad propagation of the Schrödinger or Lindblad equation to obtain P_{GS} . This pipeline was implemented with separate code, is *not* part of the software described here, and is only summarized, for completeness and not as a specification, in Sec. [S11](#).

`qanneal` was built to make step 1 auditable: every knob corresponds to a physical quantity (β an inverse temperature, Γ a transverse field, M a Trotter number, $J_{\perp}$ a Trotter bond strength), schedule endpoints are derived from the problem’s own coupling scale rather than hand-tuned, and every adaptive run exposes its full realized trajectory and observable trace.

The library provides five annealing engines (Table S1) built on a common statistical-mechanical core, and, on top of them, two schedules that pace the anneal according to a measured susceptibility rather than a fixed pre-designed ramp. One of these two schedules — the worldline-magnetization method, exposed as `method="sqa_chi"` — generalizes the worldline-susceptibility construction of Section II of the main text into a standalone, production-quality solver; Sec. S10 states precisely how this released implementation relates to the analysis that produced the main text’s own figures. The other — the bond-susceptibility method, `schedule_type="optimal"` on the `sqa/sqapt` engines — is a theoretically motivated sibling built on a different order parameter and was not used anywhere in the main text; it is described here (Sec. S8) for completeness and because its existence is independent supporting context: two distinct classically-measurable observables, $\chi_B = \text{Var}(B)$ and χ_m of Eq. (8) of the main text, are both motivated by the same finite-size-scaling argument to peak near the same critical region that exact diagonalization identifies.

Table S1: The five annealing engines provided by `qanneal`, and their role in this work.

Engine	What it simulates	Role in this work
<code>sa</code>	Classical thermal (Metropolis) annealing	Encoding sanity checks; not reported
<code>sqa</code>	Discrete-time Suzuki–Trotter path integral	Statistical-mechanical core reused by <code>sqa_chi</code>
<code>sqapt</code>	<code>sqa</code> + replica exchange on a (β, Γ) ladder	Infrastructure for larger- n extensions (Sec. S6)
<code>ctpimc</code>	Continuous-time (Trotter-error-free) path integral	Infrastructure for larger- n extensions (Sec. S7)
<code>sqa_chi</code>	Worldline-QMC with a χ_m -measured schedule	Generalized, released implementation of the main text’s surrogate method (Sec. S10)

S2 Mathematical Foundations: Ising and QUBO

Every problem `qanneal` solves reduces to one of two energy functions. The Ising form, which the main text uses throughout,

$$E(s) = \sum_{i=1}^n h_i s_i + \sum_{i<j} J_{ij} s_i s_j + c, \quad s_i \in \{-1, +1\}, \quad (\text{S1})$$

is what `DenseIsing(h, J, c)` stores for the fully-connected Sherrington–Kirkpatrick instances studied in the main text: $h = 0$ and J_{ij} drawn i.i.d. from a fixed distribution, with every pair $i < j$ coupled, which is exactly why the library’s `DenseIsing` container (dense $O(n^2)$ storage, appropriate for $n \lesssim 3000$) rather than its sparse counterpart was used. The library also accepts the equivalent Quadratic Unconstrained Binary Optimization (QUBO) form over bits $x_i \in \{0, 1\}$,

$$E(x) = \sum_{i,j} Q_{ij} x_i x_j, \quad (\text{S2})$$

and converts between the two exactly via the affine map $x_i = (1 + s_i)/2$, so that any of the classic NP-hard encodings the library documents — number partitioning, MaxCut, and penalty-constrained problems such as maximum-weight independent set — can be handed to the same solvers used for the spin-glass instances of the main text without modification.

The reason any of this is hard is frustration: cycles of bonds whose preferences cannot simultaneously be satisfied, of which the elementary example is a triangle of antiferromagnetic couplings. The Sherrington–Kirkpatrick model realizes frustration in its most extreme, all-to-all form, which is precisely why it was chosen as the benchmark ensemble for the main text — it is the setting in which the boundary-gap trap and oscillatory failure modes of Section III C of the main text are most readily provoked, and in which a schedule’s robustness to those failure modes is most stringently tested.

S3 Statistical Mechanics and the Metropolis Rule

Every engine in the library, quantum or classical, is a Markov-chain Monte Carlo sampler of a Gibbs–Boltzmann distribution driven slowly toward a zero-temperature or zero-quantum-fluctuation limit, and it is worth deriving the rule they all share, once, because every acceptance probability quoted below — for the classical spin flip, for the Suzuki–Trotter slice flip, for the parallel-tempering swap, and for the parity-parallel checkerboard sweep that measures $\chi_m(s)$ in the main text — is a special case of it.

A system with energy function E held at inverse temperature $\beta = 1/k_B T$ occupies configuration s with probability $\pi_\beta(s) = e^{-\beta E(s)}/Z(\beta)$. Sampling this distribution directly at the large β needed for optimization is as hard as the optimization problem itself, so annealing instead walks a Markov chain whose transition kernel satisfies detailed balance, $\pi(s)P(s \rightarrow s') = \pi(s')P(s' \rightarrow s)$, and slowly cools β (or, in the quantum engines, slowly raises $J_\perp$) while the chain tracks the shifting equilibrium. Factoring the transition into a symmetric proposal (pick a spin, or a slice, or a ladder rung, and propose flipping or exchanging it) and an acceptance probability A , detailed balance forces $A(s \rightarrow s')/A(s' \rightarrow s) = e^{-\beta \Delta E}$, and the choice that maximizes the acceptance rate — and hence mixing speed — is the Metropolis rule,

$$A(s \rightarrow s') = \min(1, e^{-\beta \Delta E}). \quad (\text{S3})$$

Every C++ core in the library executes exactly Eq. (S3) (or its imaginary-time generalization, Sec. S5); a cached local field $f_i = h_i + \sum_j J_{ij} s_j$ makes the energy change of a single-spin proposal an $O(1)$ evaluation, $\Delta E_i = -2s_i f_i$, and this caching is the single implementation detail most responsible for the library’s sweep throughput, including the throughput of the released `sqa_chi` pilot scan described in Sec. S9.

S4 Engine I — Simulated Annealing (sa)

The classical baseline runs Metropolis dynamics under a rising- β schedule and is not used for any result reported in the main text. It is documented here because it is the library’s simplest encoding sanity check: for a new problem class, solving small instances ($n \lesssim 20$) by `sa` and, where feasible, by brute force, is the recommended way to catch an encoding error before trusting any downstream spectral-gap or susceptibility calculation built on top of it; whether this specific check was applied to the Sherrington–Kirkpatrick instances of the main text is not verified in this supplement. Problem-adaptive tuned schedules (`auto_schedule_sa_tuned`) remove the need to hand-choose β_{start} and β_{end} : they probe the problem’s own energy scale Δ (the 75th percentile of $|\Delta E|$ over random single-spin flips) and invert the acceptance-targeting relation $\beta(p) = -\ln p/\Delta$ to hit prescribed start/end acceptance rates. This probe-and-invert pattern reappears, essentially unchanged, in the tuned schedules for every quantum engine below, including the endpoints of the pilot scan that estimates $\chi_m(s)$ in Sec. S9.

S5 Engine II — Simulated Quantum Annealing (sq_a) and the Suzuki–Trotter Mapping

This is the engine whose statistical mechanics the main text’s central observable is built from, and it is worth re-deriving the mapping in full, because Eqs. (5)–(6) of the main text are literally the output of the derivation below.

S5.1 From the transverse-field Ising model to a classical action

Quantum annealing encodes the optimization problem in the transverse-field Ising Hamiltonian $\hat{H}(\Gamma) = \hat{H}_{\text{cl}} - \Gamma \sum_i \hat{\sigma}_i^x$, where $\hat{H}_{\text{cl}}$ is diagonal in the computational basis and is exactly Eq. (S1). Real-time simulation of this Hamiltonian is exponentially expensive, so sq_a instead samples its *thermal equilibrium* at inverse temperature β via a mapping onto an equivalent classical system that Monte Carlo can handle. Trotterizing $e^{-\beta\hat{H}} = (e^{-\frac{\beta}{M}\hat{H}})^M$ and splitting each thin slice, $e^{-\frac{\beta}{M}(\hat{H}_{\text{cl}} + \hat{H}_q)} \approx e^{-\frac{\beta}{M}\hat{H}_{\text{cl}}} e^{-\frac{\beta}{M}\hat{H}_q}$, with an error of $O(\beta^2/M^2)$ per slice that vanishes as the Trotter number $M \rightarrow \infty$; inserting a complete set of classical states between every factor and using the cyclicity of the trace turns the quantum partition function into a sum over M classical replicas (“slices”) of the original spin configuration. The transverse factor between adjacent slices reduces, site by site, to a 2×2 transfer matrix $\langle s | e^{a\hat{\sigma}^x} | s' \rangle$ with $a = \beta\Gamma/M$, which can be rewritten as an ordinary classical Ising bond $\Lambda e^{J_\perp ss'}$ provided

$$J_\perp(\beta, \Gamma, M) = \frac{1}{2} \ln \coth\left(\frac{\beta\Gamma}{M}\right) = \frac{1}{2} \ln \frac{1}{\tanh(\beta\Gamma/M)}, \quad (\text{S4})$$

which is exactly Eq. (6) of the main text’s $J_\perp(\beta, \Gamma)$. Substituting back gives the effective classical action executed by the C++ core,

$$S[s] = \frac{\beta}{M} \sum_{t=1}^M E_{\text{cl}}(s^t) - J_\perp \sum_{t=1}^M \sum_{i=1}^n s_i^t s_i^{t+1}, \quad (\text{S5})$$

identical to Eq. (5) of the main text: M replicas of the classical problem, each internally coupled by the rescaled problem couplings $\frac{\beta}{M}J_{ij}$, and neighbouring replicas tied site-by-site along imaginary time by the Trotter bond $J_\perp$. Because $J_\perp$ decreases monotonically as Γ increases, an annealing run that lowers Γ from a large initial value corresponds to *raising* $J_\perp$ from near zero (decoupled, freely fluctuating slices, $\Gamma \rightarrow \infty$) to a large value (locked, classical slices, $\Gamma \rightarrow 0^+$); every adaptive-schedule interface in the library is parameterized by $J_\perp$ for this reason, and Fig. S1 sketches the lattice this action describes.

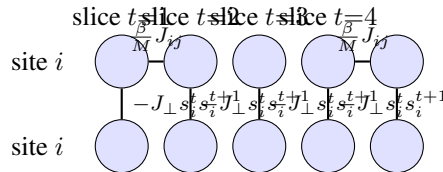


Figure S1: The Suzuki–Trotter lattice underlying Eqs. (S4)–(S5) (and Eqs. (5)–(6) of the main text): replicas of the classical problem coupled within a slice by the rescaled problem couplings, and coupled to their imaginary-time neighbours by $J_\perp$. The worldline-magnetization order parameter m of Eq. (7) of the main text is the mean spin over this entire two-dimensional lattice.

S5.2 Monte Carlo moves and why more than one move type is needed

The SQAAnnealer mixes three move types, all satisfying detailed balance for Eq. (S5): single-spin (slice) flips, whose action change has a classical part and a quantum part from the two adjacent time-bonds, $\Delta S =$

$\frac{\beta}{M}\Delta E_{\text{cl}} + 2J_{\perp}s_i^t(s_i^{t-1} + s_i^{t+1})$; whole-worldline flips, which flip one site in every slice simultaneously so that every imaginary-time bond touching it is a product of two flipped spins and therefore invariant, leaving $\Delta S_{\text{worldline}} = \frac{\beta}{M}\sum_t \Delta E_{\text{cl}}^{(t)}$ with no Trotter penalty at all; and Swendsen–Wang cluster moves along imaginary time, which grow a contiguous worldline segment with join probability $p_{\text{join}} = 1 - e^{-2J_{\perp}}$ and flip it as a unit. Single-spin flips alone become exponentially inefficient once $J_{\perp}$ is large (deep in the anneal, exactly where the susceptibility scan of Sec. S9 needs to equilibrate quickly), because $\Delta S \approx 4J_{\perp}$ for a lone slice flip; worldline flips pay no such penalty and are essential late in the anneal, while cluster moves interpolate between the two extremes and are the recommended addition (`cluster_sweeps=1`, the “+SW” variant) whenever a run visibly stalls. This mix of move types is the reason a $\chi_m(s)$ measurement equilibrates within the modest `scan_burn+scan_sweeps` budget used for the pilot scan in Sec. S9, rather than requiring the enormous sweep counts a single-spin-only sampler would need at large $J_{\perp}$.

S6 Engine III — Quantum Parallel Tempering (`sqapt`)

`sqapt` runs R complete Suzuki–Trotter systems simultaneously on a (β, Γ) ladder and periodically proposes exchanging full configurations between adjacent rungs, with a swap acceptance

$$A_{\text{swap}} = \min\left(1, \exp\left[-\left(S_r(x_{r+1}) + S_{r+1}(x_r) - S_r(x_r) - S_{r+1}(x_{r+1})\right)\right]\right) \quad (\text{S6})$$

derived from detailed balance on the joint product measure over all rungs. Expanding Eq. (S6), the swap criterion splits cleanly into a thermal part (the ordinary classical parallel-tempering criterion applied to slice-summed energies) and a quantum part proportional to $J_{\perp}^{(r)} - J_{\perp}^{(r+1)}$; when every rung shares a single evolving $J_{\perp}$, as in the shared-schedule mode of Sec. S8, the quantum part vanishes identically and the swap reduces to the classical criterion alone. `sqapt` was not used to generate any of the main text’s reported ground-state probabilities — those come from direct propagation of the time-dependent Schrödinger equation under exact diagonalization, so as to compare schedules on identical footing without introducing additional sampling noise — but it is the natural engine for the disorder-averaged, larger- n extensions the current publishability assessment of this project calls for: replica exchange is the standard remedy when read-to-read variance is large, which is exactly the symptom the main text’s Fig. 7 shows growing with n for the Roland–Cerf schedule.

S7 Engine IV — Continuous-Time Path-Integral Monte Carlo (`ctpimc`)

Taking $M \rightarrow \infty$ at fixed β in Eq. (S5) turns the discrete slice index into a continuous imaginary time $\tau \in [0, \beta)$ and each spin into a piecewise-constant worldline with kinks (domain walls in time) occurring as a Poisson process of rate Γ ; the partition function becomes $Z = \sum_{\text{configs}} \Gamma^K \exp\left[-\int_0^{\beta} E_{\text{cl}}(s(\tau)) d\tau\right]$, with K the total kink count, and the engine’s Swendsen–Wang-style cluster updates act directly on worldline segments with no discretization scale at all. `qanneal`’s implementation builds on D-Wave’s open-source `localPIMC` code (Apache-2.0, credited in the repository’s NOTICE file). This engine removes the systematic Trotter error of Sec. S5 entirely and is the natural choice if a future extension of this work were to ask whether any of the Roland–Cerf failure modes of Section III C of the main text are sensitive to the residual $O(\beta^2\Gamma^2/M)$ discretization error of `sqa` — a question the present results already answer in the negative for the parameter ranges studied (the solver-tolerance sweep of Sec. S11), but one for which `ctpimc` provides an independent, discretization-free cross-check.

S8 Adaptive Schedules I: The Bond-Susceptibility Optimal $J_\perp$ Schedule

Before describing the worldline-magnetization method that the main text uses, we describe its sibling, because the two share a derivation and their agreement is itself supporting evidence for the main text’s thesis.

S8.1 From the gap to a measurable quantity

The Roland–Cerf local-adiabaticity condition, Eq. (4) of the main text, bounds the rate of change of the control parameter by the instantaneous gap. Near a continuous quantum phase transition, finite-size scaling relates the gap to the susceptibility of the operator driving the transition, $\Delta_{\text{gap}} \sim \xi^{-z}$, $\chi \sim \xi^{2-\eta}$, so that $\Delta_{\text{gap}}^2 \sim \chi^{-2z/(2-\eta)}$. In the Suzuki–Trotter representation the natural control parameter is $J_\perp$ itself, and the natural susceptibility is that of the total imaginary-time bond sum,

$$B = \sum_{t=1}^M \sum_{i=1}^n s_i^t s_i^{t+1}, \quad \chi_B = \text{Var}(B), \quad (\text{S7})$$

the quantity that also sets the quantum part of the `sqapt` swap exponent, Eq. (S6). Substituting into the local-adiabaticity condition gives the update law

$$\Delta J_\perp = \tilde{\varepsilon} \cdot \chi_B^{-\alpha}, \quad \alpha = \frac{z}{2-\eta} + \frac{1}{2}, \quad (\text{S8})$$

with $\alpha = 15/14$ the library default for the one-dimensional quantum Ising universality class ($z = 1$, $\eta = 1/4$), a robust choice even for the mean-field Sherrington–Kirkpatrick problems of the main text because the calibration below makes the schedule’s overall pace insensitive to moderate errors in α .

S8.2 Budget calibration and the inverse-CDF trajectory

Choosing the scale $\tilde{\varepsilon}$ by hand is fragile, since χ_B grows with n , M , and the coupling distribution: a value tuned for one instance can make another leap across its whole range in a handful of steps, or — the more insidious failure — freeze, advancing $J_\perp$ so slowly that the run effectively anneals at constant coupling, a failure mode that announces itself only in the final energy, never in an error message. `qanneal` instead ties $\tilde{\varepsilon}$ to the step budget by treating Eq. (S8) as an ODE and separating variables, requiring the traversal of $[J_0, J_{\text{end}}]$ to take exactly T steps:

$$\tilde{\varepsilon} = \frac{1}{T} \int_{J_0}^{J_{\text{end}}} \chi_B(J)^\alpha dJ. \quad (\text{S9})$$

Because $\chi_B(J)$ is not known in advance, a short pilot pre-pass measures it at `calib_probes` (default 12) values of $J_\perp$ spanning the range, on scratch worldlines so the production run is unbiased, and the integral in Eq. (S9) is evaluated by the midpoint rule. Rather than integrating Eq. (S8) online — where a run that happens to order slightly faster than the pilot predicted would see an anomalously small χ_B , take an oversized step, and skip the remainder of the range — the entire trajectory is precomputed from the pilot profile via the cumulative adiabatic cost $G(J) = \int_{J_0}^J \chi_B(J')^\alpha dJ'$ and its inverse, $J(t) = G^{-1}(\frac{t}{T-1}G(J_{\text{end}}))$, placing each step at an equal increment of accumulated cost. This inverse-CDF construction is structurally identical to the cumulative time-allocation function $\tau(s)$ of Eq. (12) of the main text; the two schedules differ in which susceptibility drives the integrand and in whether the control variable is $J_\perp$ directly or the annealing parameter s , but the placement logic — spend adiabatic budget where the relevant susceptibility is large — is the same idea applied to two different observables.

S8.3 Safeguards

A single-replica guard prevents the pathological step that would otherwise occur when all within-step samples of B coincide near criticality (the naive variance collapsing to zero exactly where the step must be smallest), and a floor $\chi_B \geq 10^{-4} \max_J \chi_B^{(\text{pilot})}$ caps the step size deep in the ordered phase where $\chi_B \rightarrow 0$ and $\chi_B^{-\alpha}$ would otherwise diverge — the same regularization role played, with a different functional form, by the floor of Eq. (11) of the main text. A two-phase protocol splits the SQA run into a thermal ramp at fixed $J_\perp$ (since plain `sqa` under this schedule has no other source of thermal mobility) followed by the adaptive $J_\perp$ ramp itself; `sqapt` skips the thermal phase because its ladder and replica swaps already supply it.

S9 Adaptive Schedules II: The Worldline-Susceptibility Schedule (`sqa_chi`)

This section describes the released, generalized software implementation of the method the main text is about; Sec. S10 states exactly how it relates to the analysis that produced the main text’s own figures. Where Sec. S8’s method drives $J_\perp$ directly from the bond-sum susceptibility, `sqa_chi` is a standalone solver class, `SQACHiAnnealer`, that measures the worldline magnetization

$$m = \frac{1}{nM} \sum_{i=1}^n \sum_{k=1}^M \sigma_i^k, \quad \chi_m = nM(\langle m^2 \rangle - \langle |m| \rangle^2), \quad (\text{S10})$$

identical to Eqs. (7)–(8) of the main text, and paces the anneal in the annealing parameter s of Eq. (16) of the main text rather than in $J_\perp$ directly, via $s = A_0/(A_0 + \Gamma)$ with A_0 the driver scale.

S9.1 The pilot scan

At `scan_points` (default 16) values of s spanning the annealing interval, the solver equilibrates a freshly and independently randomized worldline for `scan_burn` (default 10) sweeps and measures χ_m over the next `scan_sweeps` (default 30) sweeps, using the parity-parallel checkerboard kernel described below. Each grid point is deliberately *not* carried forward from the previous one: a carried-over scan would gently anneal itself as it proceeds, correlating each susceptibility estimate with the point before it and making the resulting peak location harder to trust as an independent measurement. This is the same protocol — pilot scan at a sequence of s values, each from an independently randomized worldline — as the one that located $s_{\text{SQ}}^* = 0.920$ (the $n = 10$ instance) and $s_{\text{SQ}}^* = 0.876$ (the $n = 12$ instance) of the main text’s Table I, against the exact-diagonalization gap minima $s_{\text{ED}}^* = 0.883$ and 1.000 respectively; see Sec. S10 for the precise relationship between this released implementation and the code that produced those specific numbers.

S9.2 From the scan to a schedule: floor regularization and inverse-CDF placement

The scan profile is turned into a schedule exactly as Eqs. (11)–(14) of the main text describe, with one implementation specialization: the released solver applies a multiplicative floor,

$$w(s) = \max(\chi_m(s), \text{floor}), \quad \text{floor} = \max(f \cdot \max_s \chi_m(s), 10^{-12}), \quad (\text{S11})$$

with $f = \text{chi_floor_fraction}$ (default 10^{-6}), rather than the additive constant χ_0 of Eq. (11). The two play the same regularizing role, but the multiplicative floor is scale-invariant across problem sizes: because $\chi_m \sim O(nM)$, a fixed additive constant tuned for the $n = 10$ instance of Fig. 1 would be numerically

negligible for a larger ensemble member and disastrously large for a smaller one, whereas a floor defined as a fixed fraction of each instance’s own scan peak automatically rescales with the problem — a design choice intended to make the released solver usable, unmodified, across a wide range of system sizes without per-instance retuning, of the kind spanned by the $n \in \{10, \dots, 20\}$ ensemble of Figs. 7–9 of the main text (whether the original analysis code used this exact regularization for those figures is not verified here; see Sec. S10). $\chi_m(s)$ is linearly interpolated between scan points and clamped outside $[s_{lo}, s_{hi}]$; the cumulative integral $\tau(s)$ is evaluated by trapezoidal quadrature on a 4096-point grid and inverted to place each remaining step at an equal increment of accumulated weight. If the pilot scan is flat or non-finite — the `sqa_chi` analogue of the boundary-gap degeneracy discussed for Roland–Cerf in Section III C 1 of the main text — the floor falls back to 1, $w(s)$ becomes uniform, and the schedule degrades gracefully to a linear-in- s ramp rather than failing.

S9.3 The parity-parallel checkerboard kernel

Every sweep in `sqa_chi`, pilot or production, exploits a structural fact about Eq. (S5): Trotter slices of the same parity are never directly coupled, since the imaginary-time bond only ever links a slice to its opposite-parity neighbours. Held one parity fixed, every (replica, slice) cell of the other parity can therefore be updated by an independent, ordinary sequential single-spin Metropolis sweep, dispatched to a single OpenMP parallel loop; two such passes (even, then odd) constitute one full sweep of the lattice (Algorithm S1, Fig. S2). This is why `sqa_chi` parallelizes usefully even at `replicas=1`, unlike every other engine in the library, whose OpenMP parallelism is over replicas alone and is therefore inert for a single-replica run.

A tempting shortcut — flip an entire slice’s spins simultaneously, using the previous sweep’s neighbour-slice values for the in-plane coupling — is exact detailed balance only when the *spatial* coupling graph is bipartite. The Sherrington–Kirkpatrick instances of the main text are fully connected and maximally frustrated, so this shortcut would silently bias the sampled distribution for exactly the problem class studied here. The released implementation therefore uses the sequential per-slice update of Algorithm S1, which is exact for an arbitrary coupling graph while still parallelizing across slices; this is a deliberate robustness improvement made when generalizing the method for public release (Sec. S10), not a claim about the exact update rule used in the original analysis that produced the main text’s figures.

Algorithm S1 One sweep of the parity-parallel checkerboard kernel underlying `sqa_chi`

Require: Trotter lattice of R replicas $\times$ M slices $\times$ n spins; cached local fields

- 1: **for** parity $\in \{\text{even}, \text{odd}\}$ **do**
 - 2: **parallel for** every (r, t) with $t \bmod 2 = \text{parity}$:
 run a full sequential single-spin Metropolis sweep over the n spins of slice t , replica r ,
 using the classical-plus-Trotter action change of Sec. S5 and the cached local fields of Sec. S3;
 update the cache and the incremental bond sum B on every accepted flip.
 - 3: **end for**
 - 4: **return** pooled worldline magnetization m [Eq. (S10)] across all (r, t)
-

S9.4 Two-phase protocol and production anneal

As in Sec. S8, the first $\lceil \text{beta_ramp_fraction} \cdot T \rceil$ steps (default fraction 0.3) ramp β from a hot starting value to the cold target at fixed $\gamma = \gamma_{\text{start}}$; the remaining steps hold β fixed and follow the calibrated $\gamma(s_k)$ trajectory, and the solver returns its own best-energy classical configuration found along the way. Both the pilot scan and this production anneal run on independent, freshly initialized worldlines, so a `sqa_chi`

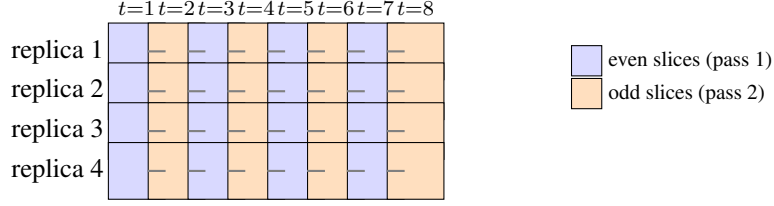


Figure S2: The checkerboard decomposition of Algorithm S1. Grey lines are the imaginary-time Trotter bonds of Fig. S1; because they always connect opposite-parity slices, one parity can be updated in a single independent parallel pass while the other is held fixed.

run's own classical result is not contaminated by state carried over from the diagnostic scan that located the critical region in the first place. Note that this classical SQA production anneal is a separate, self-contained solve — it is *not* the source of the ground-state probabilities P_{GS} reported in Figs. 3–10 of the main text, which instead come from the exact unitary evolution described in Sec. S11, using the schedule $s(t)$ that the pilot scan and the cumulative-weight construction of Eqs. (11)–(14) of the main text determine.

S9.5 Relationship between the two adaptive schedules

Table S2 summarizes the two susceptibility-paced methods side by side. Both precompute their trajectory from a pilot profile rather than integrating online, so neither is vulnerable to the compounding-noise failure discussed in Sec. S8; the choice between them is empirical, and the main text's contribution is to show that the worldline-magnetization variant, despite being the structurally simpler of the two (no universality-class exponent α to specify, no bond-sum bookkeeping), is sufficient to outperform the exact Roland–Cerf schedule on a substantial fraction of Sherrington–Kirkpatrick instances.

Table S2: The two susceptibility-paced adaptive schedules in `qanneal`.

	Optimal $J_{\perp}$ schedule	<code>sqa_chi</code> (main text)
Exposed as	<code>schedule_type="optimal"</code>	<code>standalone method="sqa_chi"</code>
Order parameter	bond sum $B = \sum_{t,i} \sigma_i^t \sigma_i^{t+1}$	magnetization $m = \bar{\sigma}$
Control variable	$J_{\perp}$ directly	s (hence Γ)
Regularizer	multiplicative floor on $\chi_B^{-\alpha}$	multiplicative floor on χ_m
Universality exponent α	required (default 15/14)	none
Pilot-scan state	carried between grid points	fresh worldline per point
Update kernel	sequential per-slice, parallel over replicas	checkerboard, parallel over replicas <i>and</i> slices
Parallel at replicas=1?	no	yes
Applicable engines	<code>sqa</code> , <code>sqapt</code> (+SW)	<code>sqa_chi</code> only

S10 Relationship Between This Release and the Main Text’s Original Analysis

The worldline-susceptibility measurement and schedule-construction algorithm of Sec. S9 (main-text Eqs. 7–15) is the same procedure, and uses the same governing equations, as the one that produced the main text’s $\chi_m(s)$ measurements, critical-point estimates, and resulting schedules (Fig. 1, Fig. 2, Table I, and the schedules underlying Figs. 3–10). It is *not*, however, literally the same code. The analysis reported in the main text was carried out with an earlier, problem-specific research script written to validate the method on the Sherrington–Kirkpatrick ensemble. The `SQACHiAnnealer` class and `method="sqachi"` solver described in this supplement are a subsequent, generalized C++/Python reimplementaion of that same method, prepared for the public release of `qanneal` so that the method is usable on arbitrary dense or sparse Ising and QUBO problems rather than only the SK instances studied here. Concretely:

- The governing equations (worldline magnetization and susceptibility, Eq. (S10); the annealing-parameter mapping $s = A_0/(A_0 + \Gamma)$; the cumulative-weight schedule construction) are identical between the original analysis and the released solver.
- The released solver’s update kernel (the parity-parallel checkerboard sweep, Sec. S9.3 below and Algorithm S1) generalizes the update rule used in the original analysis to arbitrary, non-bipartite coupling graphs; for the fully-connected Sherrington–Kirkpatrick instances of the main text the two are expected to sample the same equilibrium distribution but are not bit-for-bit identical programs.
- The released solver’s regularization (the multiplicative floor of Eq. (S11) below) and its independently-re-randomized pilot scan are engineering choices made for robustness across arbitrary problem sizes; whether the original analysis used precisely these choices or close variants of them was not re-verified for this supplement.
- Table S4 therefore reports the released solver’s *default* configuration for this method, not a verified record of the exact parameters used to produce any specific main-text figure.

We report the released implementation in full, rather than only the original research script, because it is the version intended for reuse and independent verification of the method itself; readers wishing to exactly reproduce a specific main-text figure bit-for-bit should treat this supplement as a specification of the *method*, not as the literal analysis code.

S11 Numerical Protocol for the Reference Roland–Cerf Schedule

For completeness, we document here the numerical protocol used to generate the exact Roland–Cerf reference schedule and ground-state probabilities against which the library’s surrogate is judged, since these were produced outside `qanneal` proper (by direct exact diagonalization and time-dependent Schrödinger-equation propagation) and their reliability is essential context for interpreting every figure in Section III of the main text. Spectral data ($\Delta(s)$ and the transition matrix element of Eq. (4)) were obtained for each disorder realization by full exact diagonalization of $H(s)$ on a dense grid of s (400–800 points, refined adaptively near the minimum gap); ground-state success probabilities were obtained by adaptive-step Runge–Kutta (embedded Dormand–Prince) integration of the Schrödinger equation under each schedule, with tolerances swept over 10^{-8} – 10^{-12} and the schedule’s own internal discretization varied independently, in order to test the oscillatory Roland–Cerf failure mode of Section III C 2 of the main text for solver artifacts; the oscillation period and amplitude were unchanged to within machine precision across this entire sweep. The

Lindblad robustness check of Section III F integrates $\dot{\rho} = -i[H(s(t)), \rho] + \sum_k \gamma \mathcal{D}[\sigma_z^{(k)}]\rho$ for the same representative instance at dephasing rates $\gamma \in \{0, 10^{-3}, 10^{-2}, 5 \times 10^{-2}\}$ with a fixed-step fourth-order Runge–Kutta propagator of the vectorized Liouvillian.

S12 Parallel and HPC Deployment

This section describes the released library’s parallel-execution capability, of the kind suited to disorder-averaged ensembles such as Figs. 7–9 of the main text (20–100 independent Sherrington–Kirkpatrick instances per system size, across six system sizes); it is not a record of the specific hardware or job configuration used for those particular figures (Sec. S10). `qanneal` supports three nested levels of parallelism: OpenMP threads within a single `sqa_chi` pilot scan or run (parallel over replicas, and, uniquely for this engine, over Trotter slices as well, Sec. S9); independent worker processes distributing reads and disorder realizations across the cores of a single node; and SLURM job arrays distributing realizations across nodes of an HPC cluster (the `nandadevi` and `kamet` clusters at IMSc were used for other benchmarking of this library, not verified here as the specific resource behind any main-text figure), one array task per disorder seed, mergeable afterward into ensemble statistics of the form of Eq. (S12) below. The job-array route is preferred by the library’s own documentation over a long-running MPI job for the usual reason: a failed task costs one disorder realization, not the whole ensemble. Reproducibility of a `sqa_chi` run only requires recording the random seed and the solver configuration (cf. Table S4 for the released defaults); OpenMP thread count and process count do not affect the result, since each parallel unit owns a private RNG stream, local-field cache, and cluster buffers.

S13 Concordance: Main-Text Results and the Methods Behind Them

Table S3 maps every quantitative result in Sections II–III of the main text to the *method* responsible for it, and states plainly whether that method is one this supplement documents in software form (worldline-susceptibility measurement and schedule construction, Secs. S5–S9) or one that is outside its scope (exact diagonalization, unitary/Lindblad evolution, disorder averaging over independently generated realizations). Per Sec. S10, entries in the third column naming `sqa_chi` or `SQACHiAnnealer` refer to the method and its released, generalized implementation, not a verified record of the literal code that ran for the main text.

$$\text{SEM}(T) = \frac{1}{\sqrt{N_{\text{inst}}}} \sqrt{\frac{1}{N_{\text{inst}} - 1} \sum_{a=1}^{N_{\text{inst}}} (P_{\text{GS}}^{(a)}(T) - \overline{P}_{\text{GS}}(T))^2} \quad (\text{S12})$$

Table S4 records the released `sqa_chi` solver’s *default* configuration for this method (Sec. S10); parameters not listed were left at their library defaults (Sec. S14).

S14 Condensed API Reference

The library exposes a single high-level entry point, `solve(problem, method=..., **kwargs)`, which auto-detects the problem type (`DenseIsing`, `SparseIsing`, `QUBO`, `NumPy array`, `dict/list`, `dimod BQM`, or `networkx graph`), auto-tunes a schedule from the problem’s own coupling scale when none is supplied, runs `reads` independent restarts, and returns the best result. Table S5 condenses the parameters most relevant to reproducing this work; the full reference, including every observer and result class, is distributed with the repository.

Table S3: Concordance between main-text results and the methods that produced them.

Main-text item	Equation(s)	Method (software scope, if any)
Fig. 1, Table I (s_{SQ}^*)	Eqs. (7)–(9)	Worldline-susceptibility pilot scan; released as <code>SQACHiAnnealer</code> (Sec. S9)
Fig. 2 (schedule shapes)	Eqs. (11)–(15)	Cumulative-weight inverse-CDF construction; released as <code>sqa_chi</code> (Sec. S9)
Figs. 3–10 (P_{GS} vs. T)	Eqs. (18)–(19)	Exact diagonalization + TDSE propagation (<i>not</i> part of this software); driven by the critical-region estimate of Fig. 1 (Sec. S11)
Fig. 5, Eq. (20) (RC cumulative allocation)	Eq. (20)	Exact-diagonalization $\Delta(s)$; outside this software’s scope
Fig. 6 (RC oscillation)	—	Solver-tolerance sweep on the TDSE propagator; outside this software’s scope (Sec. S11)
Figs. 7–9 (disorder averages)	Eq. (S12)	Ensemble statistics over independently generated SK realizations; outside this software’s scope
Fig. 12 (failure-class fractions)	—	Classification protocol applied to the Figs. 7–9 ensemble; outside this software’s scope
Fig. 13 (Lindblad)	Eq. (21)	Lindblad propagator; outside this software’s scope (Sec. S11)
Figs. 14–15 (β , M robustness)	—	Worldline-susceptibility pilot scan swept over M, β ; released as <code>SQACHiAnnealer</code>

S15 Symbol Glossary

S16 Code and Data Availability

`qanneal` v0.7.0 is released under the Apache License 2.0 and includes, alongside the five engines and two adaptive schedules described above: a complete `pybind11`-bound C++17 core; a Python front end with auto-tuned schedules and problem adapters for NumPy, dict/list, `dimod`, and `networkx` inputs; observer classes that capture per-sweep energy, magnetization, and full spin-state traces for every engine; OpenMP, multiprocessing, and MPI/SLURM parallelism (Sec. S12); and a pre-flight sanity script, `scripts/sanity_schedule.py`, that runs the *bond-susceptibility* adaptive schedule (`schedule_type="optimal"`, Sec. S8, not the worldline-susceptibility method of the main text) on small reference instances and classifies the realized $J_{\perp}(t)$ trajectory as PASS, DEGENERATE, or FAIL before any large-scale compute is spent on that schedule — included here as an example of the library’s testing conventions, not as part of the main text’s own pipeline. The complete derivations of every equation in this supplement, from the Metropolis rule of Sec. S3 through the C++ implementation notes for contributors, are given from first principles in the library’s companion manual, distributed with the source under `docs/manual/`.

Table S4: Default `sqa_chi` solver configuration in the current `qanneal` release (Sec. S10).

Parameter	Value
<code>trotter_slices</code> (M)	32
<code>replicas</code>	4
<code>optimal_num_steps</code> (step budget T)	150
<code>optimal_beta_ramp_fraction</code>	0.3
<code>chi_scan_points</code>	16
<code>chi_scan_sweeps</code>	30
<code>chi_scan_burn</code>	10
<code>chi_floor_fraction</code>	10^{-6}
<code>chi_driver_A0</code>	≤ 0 (resolves to <code>chi_gamma_start</code>)
<code>sweeps_per_step</code>	20

Table S5: Condensed `solve()` parameter reference (see repository documentation for the complete list).

Parameter	Meaning
<code>method</code>	"sa", "sqa", "sqapt", "ctpimc", "sqa_chi"
<code>reads</code>	Independent restarts; best over all reads returned
<code>trotter_slices</code>	Imaginary-time slices M (Sec. S5)
<code>replicas</code>	Parallel Trotter systems (sqa) or ladder rungs (sqapt)
<code>worldline_sweeps</code> , <code>cluster_sweeps</code>	Whole-worldline and Swendsen–Wang move counts (Sec. S5)
<code>schedule_type</code>	"standard" or "optimal" (Sec. S8)
<code>optimal_eps_tilde</code> , <code>optimal_alpha</code> , <code>optimal_num_steps</code>	Bond-susceptibility schedule parameters (Eqs. (S8)–(S9))
<code>chi_scan_points</code> , <code>chi_scan_sweeps</code> , <code>chi_scan_burn</code> , <code>chi_floor_fraction</code>	<code>sqa_chi</code> pilot-scan parameters (Sec. S9)
<code>seed</code>	RNG seed; fixed seed + fixed replica count $\Rightarrow$ bitwise-reproducible runs

Table S6: Symbol glossary (consistent with the main text and this supplement).

Symbol	API name	Meaning
$s_i \in \{-1, +1\}$	<code>spins</code>	Ising variable
h_i, J_{ij}, c	<code>h, J, c</code>	local field, coupling, energy offset
β	<code>beta</code>	inverse temperature
Γ	<code>gamma</code>	transverse-field strength
M	<code>trotter_slices</code>	number of imaginary-time slices
$J_\perp$	<code>j_perp</code>	Trotter coupling, Eq. (S4)
B, χ_B	—	bond sum and its variance, Eq. (S7)
m, χ_m	—	worldline magnetization and its susceptibility, Eq. (S10)
$\tilde{\epsilon}$	<code>eps_tilde</code>	adiabaticity scale, Eq. (S9)
α	<code>alpha</code>	universality exponent, Eq. (S8)
s	—	annealing parameter, $s = A_0/(A_0 + \Gamma)$