Ladder-Graded Spacing: execution-aware scheduling for code practice

DeepForge Research

September 18, 2026 · lgs

Technical report · Not peer reviewed

Abstract

Spaced-repetition schedulers for coding practice see more than a pass or a fail: DeepForge observes how many test runs failed before the pass, which hint tier was revealed, and whether the learner reset the buffer before succeeding. The current SM-2 path collapses that ladder to a single quality value and discards the rest. This paper introduces Ladder-Graded Spacing (LGS), a fixed-parameter scheduler that keeps the execution ladder as graded evidence: failed runs, hint tiers, and resets map to an effective failure count phi, which drives observed difficulty D and an effort EWMA, while scheduling uses the FSRS-6 power-law stability update at a 0.9 target retention. An ablation showed the originally proposed evidence multiplier E(phi) cost 27% more reviews with no Brier gain, so the shipped variant (noE) keeps the ladder in D and in the effort-driven interval cap only. In a 100-learner, 40-item, 365-day simulation against a memory-based truth model, noE used 25.63% fewer reviews per retained item than SM-2 at matched first-attempt recall (25.32% in the literal protocol), with a Brier delta of -0.1105 and no seed worse; 30/30 seeds passed the 10% gate. The paper documents the final model, the migration guard that protects existing state, the simulation protocol, and the threats to validity, including that hint debits have no published calibration.

Keywords: spaced repetition, scheduling, FSRS-6, SM-2, retrievability, execution signals, code practice

---

1. Introduction & motivation

DeepForge is a practice platform where learners implement machine-learning and math building blocks in Python from scratch and receive immediate, execution-verified feedback in the browser. Every solved problem enters a spaced review queue: on a due date the learner re-solves the problem from scratch, and the scheduler decides when it returns. That queue is the one place where the platform applies memory science on a schedule of its own rather than the learner's.

The queue inherits from SM-2. A session is graded by qualityFromRun, collapsed to one of 0/3/4/5, and turned into an ease factor and an interval stored as a single record per problem. The collapse throws away signals the platform already owns: how many test runs failed before the pass (failedRuns), which hint tier was revealed (hintTier in 0..3), and whether the code buffer was reset before the pass (resetBeforePass). In the current path the reset flag is not even passed from the problem view on an ordinary solve, so a hint-heavy session after three failures and a clean first-try solve can land on nearly the same ease and interval.

The motivation is not that self-report is dishonest — the platform never asks the learner to self-grade a solve. It is that an execution ladder is objective evidence about difficulty and effort, and the three-value collapse discards it. Failed test runs are observable errors, a stronger signal than the response-time proxies now entering scheduling research, and hint tiers are earned by a budget

rather than offered on request: tier 2 requires one failed run or three minutes of struggle, tier 3 requires two failed runs or six minutes. A hint that deep in the session records how hard retrieval actually was.

This paper specifies Ladder-Graded Spacing (LGS), the technique invented in DeepForge research wave 40, and reports its measured behavior. LGS keeps the ladder and changes the model around it: a ladder observation turns the raw signal into an effective failure count phi; phi updates an observed difficulty D and an effort EWMA; success growth uses the FSRS-6 long-term stability formula; failure uses the FSRS-6 post-lapse stability instead of resetting to one day; and an effort-driven cap bounds the interim interval for repeatedly struggling items. Queue order and the readiness curve move to the same quantity, predicted retrievability R(t, S) under a power-law forgetting curve.

- Execution evidence: define phi from failed runs, hint tier, and reset, with explicit clamping for junk input.
- Fixed-parameter memory model: FSRS-6 defaults, no per-user fitting, no network, no randomness, no new dependencies.
- Ladder channel: phi updates D and effort; in the shipped variant it does not multiply stability.
- Migration guard: existing SM-2 records upgrade in place with due dates byte-identical and idempotent output.
- Simulation: a deterministic, two-run-identical protocol with paired-by-learner bootstrap gates, plus edge-learner cohorts.

> *Scope — LGS changes scheduling, ordering, and readiness only. Problem content, hint budgets, grading, and the review surface are unchanged. It supersedes hypotheses H1 (overdue-ratio ordering), H2 (linear readiness curve), and H4 (lapse-count leech threshold from the learning-science review); H3 and H5 remain out of scope and compose with it.*

2. Related work

Spacing is one of the most replicated effects in learning research, but the shape of the schedule is contested. Cepeda et al. (2006), across 839 assessments, found roughly a 15% average benefit for long retention intervals and a non-monotonic lag function: for a given retention interval there is an optimal inter-study gap, and the optimal gap grows with the retention interval. Expanding versus fixed is not settled in favor of expanding: Karpicke & Bauernschmidt (2011) found absolute spacing drove large gains while expanding, equal, and contracting relative schedules were indistinguishable; Latimier et al. (2021) found no expanding-versus-uniform difference (g = 0.034, n.s.) across 54 effect sizes; Kang et al. (2014) found expanding equal at an eight-week final test; Storm et al. (2010) found expanding wins when interference is high. Task complexity moderates the effect: Donovan & Radosevich (1999) report d = 0.46 overall but near-zero for high-complexity tasks (0.07-0.11) versus 0.97 for psychomotor tasks, so spacing should help a long interview problem, but by less than the flashcard literature implies.

FSRS models difficulty, stability, and retrievability (DSR) and schedules to an explicit desired retention. FSRS-6 (21 weights, optimizable forgetting-curve decay) ships in Anki since 25.07 (Anki PR 3929); FSRS-7 (35 weights, fractional intervals, same-day predictions) merged into the

benchmark in March 2026, and the fsrs-rs integration PR merged on 2026-09-08. The original algorithm paper is Ye et al., KDD 2022, and a browser-ready TypeScript implementation (ts-fsrs) exists for the web stack. Calibration gains are large and replicated: on about 350 million held-out reviews, FSRS-6 log loss is 0.346 versus 0.436 for FSRS v3; on the earlier 20,000-collection benchmark, SM-2 scored 0.73 log loss and 0.21 RMSE(bins) versus FSRS v4 at 0.39 and 0.046 (Expertium breakdown), and the corrected FSRS-7 benchmark figure is 0.3401 against 0.3620 for default parameters. The benchmark authors themselves caution that SM-2 never predicted probabilities, so converting its intervals to probabilities requires assumptions; the comparison is directionally strong, not perfectly fair.

Those numbers are flashcard recall, and execution-verified multi-minute problems are out of distribution for them. Community simulations report 20-30% fewer reviews for the same retention, and per-user optimization beats default parameters for roughly 84% of collections, but the Anki manual warns optimization needs at least a few hundred reviews to generalize. DeepForge stores one state row per problem and has no per-review log, which is exactly what an optimizer needs. FSRS's own benchmark shows default parameters give up much of the edge of optimized ones, so LGS is honest about being fixed-parameter and must be validated by simulation, not assumed.

Attempt-level frequency models support the shape LGS borrows. DASH and DAS3H count correct and attempted responses with log transforms — diminishing returns, the power law of practice — and beat IRT baselines on tutoring data. HLR predicts recall about 45% better than Leitner, but needs hundreds of traces per item, which DeepForge does not have. Bayesian and deep knowledge tracing are rejected here for the same data reason (Corbett & Anderson 1994; Piech et al. 2015): per-step tracing needs step logs or large samples per skill, and LGS stays at review-event level. The lesson taken is narrower: more failures should earn less credit, monotonically, with a fixed floor.

Objective outcome cues are entering scheduling research. ARTS used response time plus accuracy and beat fixed spacing at both immediate and delayed tests; SSP-MMC++ reports a 22.8% improvement in target-half-life achievement and a 16.8% reduction in retention cost. But step-level response-time propensities are ambiguous and proficiency-moderated, and DeepForge does not persist time per problem anyway. LGS therefore uses errors and hints — both persisted and both objective — and explicitly avoids wall-clock features.

Unsuccessful retrieval followed by immediate feedback teaches something: Kornell, Hays & Bjork (2009) and Hays et al. (2013) found that failed retrieval with immediate feedback enhances later learning. Retrieval practice itself is strong evidence (Rowland 2014 reports g around 0.50, stronger for effortful recall), but difficulty has a floor: it helps only when the learner has the prior knowledge to succeed, otherwise it is an undesirable difficulty (Bjork & Bjork 2020). In-app feedback is immediate and execution-verified, so a struggled pass is discounted by LGS but never treated as a lapse. This is also why the ladder is legitimate memory evidence rather than a punishment.

Assistance is not free. Hint shortcuts correlate with worse learning (Stamper, LAK 2026), hint use improves help-need prediction (Maniktala 2022), and the assistance dilemma (Koedinger & Aleven 2007) is well documented. The evidence is weak in one specific place, and LGS states it plainly: nothing published maps hint tiers or failed runs to a stability multiplier. The hintDebit values in LGS are chosen, swept in simulation, and first to cut if calibration fails.

Item difficulty from outcomes is usually estimated with Elo-style updates that approximate IRT quality

(Pelánek 2016). LGS uses a single exponential update because there is one local user and no matchmaking; that is weaker than a fitted IRT model and is flagged as such. Finally, leech thresholds are lore: Anki practice tags and suspends after a configurable lapse count (Anki manual), but no controlled trial validates the threshold or the actions, and FSRS-era statistical leech detection remains a design discussion rather than shipped behavior.

Source	LGS takes	LGS does not take
Spacing literature	Absolute gaps grow; fixed schedule is defensible	An expanding-specific advantage
FSRS-6	Power-law curve, success and lapse stability, 0.9 target	Per-user weights (no log, no fitting)
DASH / DAS3H	Monotone diminishing credit for more failures	Log-transformed counts fitted to tutoring data
Response-time models	Objective outcome cues matter	Wall-clock features (time is not persisted)
Feedback research	Struggled pass is discounted, not failed	Treating a struggled pass as a lapse
Leech lore	Repeated struggle should shorten intervals	The specific Anki lapse threshold

What LGS borrows and what it rejects

3. Method

LGS is a pure function of stored state, an execution signal, and an injected clock. It has no dependence on wall-clock time, randomness, the network, or any new package, so the same inputs produce identical output on every run. The grade-time signal is passed flag, failed runs (an integer, at least zero), hint tier in 0..3, and a reset flag.

The ladder observation reduces the signal to an effective failure count phi. Failed runs and a reset form the retrieval part, with the reset worth 0.5 of a failure because a reset may be refactoring rather than forgetting. The hint part adds a per-tier debit. Every input is clamped before it touches arithmetic: hint tier is floored and clamped to 0..3, failed runs are non-negative integers, and non-finite values read as zero, so junk input can never propagate NaN into a due date.

```
                    Effective failure count
phi = min(fMax, f + (reset ? wReset : 0)) + hintDebit[clamp(0, 3,
                        floor(hintTier))]
```

fMax = 12 caps the failed-run part only, so phi maxes out at 16 with the tier-3 debit of 4.

Memory is represented by a power-law forgetting curve with FSRS-6's default decay and a stability S measured in days at a 0.9 retrievability target. The factor F is derived, not fitted, so that R(S, S) = 0.9 exactly; the interval that returns to 0.9 is exactly S, which keeps the target retention semantic and makes intervals and stabilities interchangeable.

```
                  Retrievability and interval
      R(t, S) = (1 + F * t / S)^(-d), F = 0.9^(-1/d) - 1 =
0.9803464944134797, d = 0.1542; I(r, S) = (S / F) * (r^(-1/d) - 1)
```

I(0.9, S) = S; with the 0.9 target, the next interval rounds S'.

On a pass, stability grows with the FSRS-6 long-term formula. This is the shipped variant, called noE: the ladder is deliberately absent from the stability multiplication. On a failure, stability uses the FSRS-6 post-lapse formula, which preserves part of a mature item's interval rather than resetting to one day, while still scheduling weak items soon. Both branches then update difficulty and effort from phi.

```
Success stability (noE)
S' = S * (1 + exp(w8) * (11 - D) * S^(-w9) * (exp(w10 * (1 - rPred)) -
                                    1))
```

rPred = R(elapsed, S), the pre-review prediction. Defaults: w8 = 1.8722, w9 = 0.1666, w10 = 0.796.

```
Post-lapse stability
S' = w11 * D^(-w12) * ((S + 1)^(w13) - 1) * exp(w14 * (1 - rPred))
```

Defaults: w11 = 1.4835, w12 = 0.0614, w13 = 0.2629, w14 = 1.6483. At D = 5, phi = 0, rPred = 0.9, this yields S' = 0.316769 / 1.391987 / 3.747287 for S = 1 / 10 / 100, with intervals 1 / 1 / 4.

Difficulty is an observed value: the model computes what difficulty the execution ladder implies and moves the stored D toward it with a fixed learning rate. In the noE variant this is the ladder's main channel on passes. Effort is an EWMA of normalized struggle that persists across reviews: a failed review contributes a larger increment than a pass, and any clean pass halves the accumulator. When accumulated effort reaches the cap, the next interval is bounded to one week, so repeated struggle cannot buy a long absence. The cap is entered by lapses or repeated high-phi reviews, not by a single struggled pass: from effort = 0, even a maximal phi = 16 review only reaches effort' = 0.5.

```
Difficulty and effort update
D_obs = pass ? 3 + 7 * phi / (phi + 6) : 10; D' = clamp(1, 10, D + eta
  * (D_obs - D)); e = clamp(0, 1, (phi + (pass ? 0 : wForget)) / 8);
                  effort' = 0.5 * effort + 0.5 * e
```

If effort' >= 0.6 the interval is capped at 7 days. A clean pass (e = 0) halves effort and releases the cap.

Scheduling then writes the standard fields. Elapsed days come from calendar date keys, never wall-clock deltas: the difference from lastReviewedAt when present, otherwise the stored interval plus any days past due. This matters for determinism across time zones and daylight-saving transitions; the specification's original elapsedDays was undefined and changed results by roughly a factor of three until it was pinned to date keys.

```
Scheduling write-back
S' = clamp(0.1, 36500, S'); interval = clamp(1, 36500, round(I(0.9,
    S'))); if effort' >= 0.6 interval = min(interval, 7); due =
addDays(todayKey, interval); reps' = pass ? reps + 1 : 0; lapses' =
                  pass ? lapses : lapses + 1
```

Queue order and the readiness curve read the same model. Due items sort by predicted retrievability ascending, so the most-forgotten item is first, with the existing weak-category, due, difficulty, and id tiebreaks after that, then the unchanged category interleaving and a cap of ten.

Readiness uses R(t, S) for scheduled items and R(t, S0) with S0 = 2.3065 (the FSRS-6 initial Good stability) for unresolved items; the 0.7 retained threshold is unchanged.

Parameter	Value	Provenance
decay	0.1542	FSRS-6 w20 default
w8, w9, w10	1.8722, 0.1666, 0.796	FSRS-6 defaults (success growth)
w11..w14	1.4835, 0.0614, 0.2629, 1.6483	FSRS-6 defaults (post-lapse)
target retention	0.9	FSRS/Anki default; I(0.9, S) = S
fMax	12	Chosen; caps the failed-run part only, phi <= 16
wReset	0.5	Chosen; a reset may be refactoring, not forgetting
hintDebit	[0, 0.5, 1.5, 4.0]	Chosen; feeds D_obs and effort only (no published calibration)
wForget	4	Chosen; effort increment for a failed review
eta	0.15	Chosen; difficulty learning rate, swept [0.05, 0.3]
seed S0, D0	1, 5	S0 so the due=+1 seed has R = 0.9; D0 from FSRS-0
effortCap, capInterval	0.6, 7 days	Chosen; lapse/struggle-entered cap, weakest default
bounds	S in [0.1, 36500], interval in [1, 36500]	Anki max-interval default; floor chosen
removed	kappa = 0.7, phi0 = 6, E(phi)	Ablation: +27% reviews, no Brier gain

Parameters. Final model parameters and their provenance

Existing records upgrade through a guarded migration that runs on every read but only derives what is missing. Stability, difficulty, and effort are derived from the legacy fields when the record is not version 2, or when the specific field is missing, non-finite, or out of range; valid version-2 fields are copied verbatim. Everything else — due date, interval, reps, lapses, lastGrade, lastReviewedAt, and the frozen ease rollback field — is preserved exactly. The output key order is fixed, so migrating twice is byte-identical to migrating once, and a stale version-1 record that wins a merge is re-migrated on the next read. No storage key, backup inventory, or merge rule changes; the new fields travel atomically with the record under the existing last-write-wins rule, and field-level merging of S, D, or effort is explicitly forbidden because max and average are wrong for a sequential update.

```
Migration derivation (v1 records only)
S = reps > 0 ? clamp(0.1, 36500, interval) : 1; D = clamp(1, 10, 5 +
 (2.5 - ease) * 2); effort = {0: 0.9, 3: 0.6, 4: 0.3, 5: 0.1, null:
                          0}[lastGrade]
```

Example: ease 1.7, interval 14 gives D = 6.6. No due date changes on upgrade; the next grade recomputes it.

The grade-time pipeline (mirrors src/lib/lgs.ts)

```
const phi = ladderPhi(signal, params);
const rPred = lgsRetrievabilityAt(elapsed, S, params);
const rawS = signal.passed
```

```
  ? S * (1 + Math.exp(w8) * (11 - D) * Math.pow(S, -w9) * (Math.exp(w10 * (1 -
rPred)) - 1))
  : w11 * Math.pow(D, -w12) * (Math.pow(S + 1, w13) - 1) * Math.exp(w14 * (1 -
rPred));
const nextS = clamp(0.1, 36500, rawS);
const dObs = signal.passed ? 3 + (7 * phi) / (phi + 6) : 10;
const nextD = clamp(1, 10, D + eta * (dObs - D));
const e = clamp(0, 1, (phi + (signal.passed ? 0 : wForget)) / 8);
const nextEffort = 0.5 * effort + 0.5 * e;
let interval = clamp(1, 36500, Math.round(lgsIntervalFor(nextS)));
if (nextEffort >= effortCap) interval = Math.min(interval, capInterval);
return { ...next, due: addDays(todayKey, interval), S: nextS, D: nextD, effort:
nextEffort };
```

> *The evidence multiplier was retired — The original spec multiplied success stability by E(phi) = (1 + phi/6)^(-0.7), which is 1 at phi = 0 and 0.5 at phi = 6, with a floor of (11/3)^(-0.7) = 0.40273 at phi = 16. Simulation showed the multiplier costs +27% reviews with no Brier gain (the lgsE1 minus lgs Brier confidence interval includes zero), so the spec's own kill rule retired it, along with its kappa and phi0 parameters. The shipped noE variant never shrinks stability on a pass: its measured minimum pass ratio is 1.000900.*

4. Simulation & results

Because there is no per-review log and no external dataset of execution-graded code practice, LGS was validated in a deterministic simulation rather than on user data. The cohort is 100 learners, 40 items, and a 365-day horizon, extendable once to 400 learners if a result is inconclusive. True ability is drawn a ~ N(0, 0.7^2), true item difficulty delta ~ N(0, 1), and true initial stability S* ~ lognormal(0.8, 0.5) clipped to [0.3, 20]. A synthetic hint policy mirrors the real one: tier 2 after one failure, tier 3 after two. Attempts are Bernoulli with a per-session probability, up to four failures per session.

The protocol was corrected during verification. The original ground truth used the full session pass probability in the memory update, which made the literal protocol degenerate: every arm retained 4000/4000 items and the comparison carried no information. The corrected truth model updates memory from memory retrievability alone, while the session pass probability multiplies in an ability offset, p = Rmem * sigma(a - delta). Acceptance uses the memory-only truth model with ability-offset interpolation at first-attempt recall 0.5; the literal numbers are reported for continuity but are not the basis for shipping.

Three arms are compared. The lgs arm is the shipped noE scheduler. The sm2 arm is the exact current code path: qualityFromRun plus the hint penalty plus the existing grade function. The oracle arm knows the true stability S* and serves as a ceiling, not a competitor. The primary metric is reviews per item still retrievable at day 365. Each arm's pre-review prediction is also scored with Brier loss; SM-2's interval is converted to a probability by the same curve, because SM-2 never emitted probabilities. Secondary metrics are first-attempt recall, the overdue-failure rate, and p95 reviews per day.

- Cohort: L = 100 learners (extend to 400 once), M = 40 items, 365 days.
- Truth: memory-only update with ability-offset session probability; hints tier 2 after one failure, tier 3

after two.

- Arms: lgs (noE), sm2 (current path), oracle (true S*, ceiling only).
- Stopping: paired-by-learner bootstrap, 2,000 fixed-seed resamples; stop when the 95% CI of LGS minus SM-2 excludes zero, else extend once, else declare inconclusive.
- Gates: at least 10% fewer reviews at matched recall, Brier no worse than SM-2 by more than 0.01, no edge cohort above 1.10x SM-2 reviews, and two runs byte-identical.

The default spec with the E multiplier failed the primary gate: 3.10% fewer reviews, 0 of 30 seeds at or above 10%. Lowering the tier-3 hint debit to 1.5 passed (14.31% across seeds), and the ablation attributed the E cost to +27% reviews with no calibration gain. That is why the chosen variant drops E: it clears the gates broadly, improves calibration, and keeps every edge cohort inside the 1.10x bound.

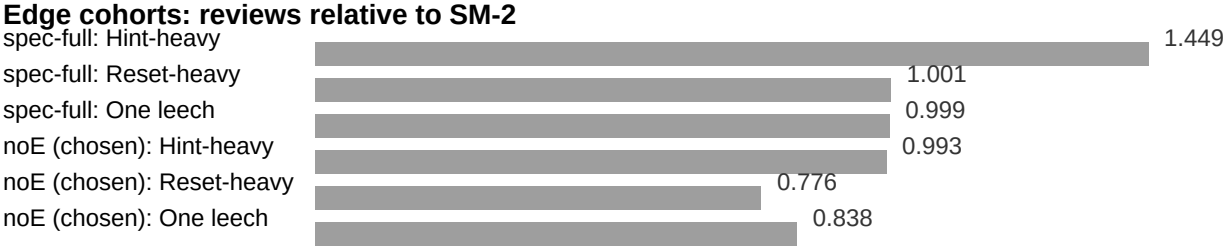
Variant	Literal % fewer	Matched % fewer	30-seed CI	Seeds >= 10%
spec-full (with E)	3.10	18.22	2.85 [2.32, 3.37]	0/30
retuned h15	15.57	20.94	14.31 [13.75, 14.87]	30/30
noE (chosen)	25.32	25.63	24.20 [23.70, 24.70]	30/30
nonHintE	16.38	23.01	14.79 [14.29, 15.29]	30/30

Variant ablation (L = 100, M = 40, 365 days). Literal is the paper protocol with ground-truth pass probability p; matched is the memory-only truth model with ability-offset interpolation at recall 0.5. The 30-seed interval is the mean and 95% bootstrap CI of the matched percent-fewer metric.

Cohort	spec-full	retuned h15	noE (chosen)	nonHintE
Hint-heavy	1.449	0.904	0.9933	1.074
Reset-heavy	1.001	0.880	0.7755	0.887
One leech	0.999	0.919	0.8375	0.917

Edge learners: reviews relative to SM-2 (lower is better). The ship gate is <= 1.10x SM-2 reviews for every edge cohort.

Edge cohorts: reviews relative to SM-2



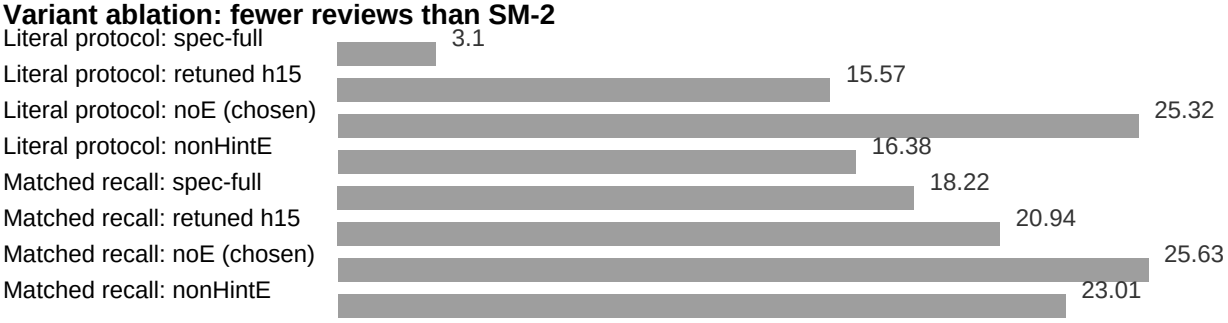
Reviews per retained item relative to SM-2 (1.00 is parity, lower is better) for the three edge cohorts, spec-full versus the shipped noE variant. The ship gate is 1.10x; spec-full breaks it on hint-heavy (1.449x), while noE finishes below parity in every cohort (0.9933 / 0.7755 / 0.8375). The retuned h15 and nonHintE arms are in the edge table above.

Metric	Result
Literal protocol	25.32% fewer reviews than SM-2
Matched recall 0.5	25.63% fewer reviews
30-seed CI (matched)	24.20 [23.70, 24.70]; 30/30 seeds >= 10%
Memory-truth CI	20.84 [20.41, 21.26]; 30/30 seeds

Metric	Result
Brier delta vs SM-2	-0.1105 [-0.1129, -0.1081]; 0/30 seeds worse
Raw recall	-2.53pp [-2.57, -2.50] (memory-truth reading)
All-fail cohort	1.000x SM-2 reviews
Perfect cohort	0.833x SM-2 reviews
p95 reviews/day	442

Chosen configuration (noE): headline outcomes

Variant ablation: fewer reviews than SM-2



Percent fewer reviews per retained item than SM-2 for four variants. The dashed line is the 10% ship gate. The literal protocol (left bar of each pair) is degenerate — every arm retains 4000/4000 — so acceptance uses the matched-recall comparison (right bar), where the chosen noE variant leads at 25.63% and passes the gate on 30/30 seeds.

The chosen configuration's 30-seed confidence interval for matched percent-fewer reviews is 24.20 [23.70, 24.70], with 30/30 seeds clearing the 10% gate. The memory-truth interval is 20.84 [20.41, 21.26], also 30/30. Brier loss improves by 0.1105 on average (CI [-0.1129, -0.1081]) and no seed is worse; raw recall is 2.53 percentage points lower (CI [-2.57, -2.50]) on the memory-truth reading, which is the expected trade when fewer reviews are spent. The degenerate cohorts behave sensibly: an all-fail learner costs 1.000x SM-2 reviews and a perfect learner 0.833x, while the stress cohorts that nearly killed the spec-full variant are now comfortably below baseline.

The ladder itself is visible in the ablation. The survey of the 40x40x180-day smoke probe pointed the intended direction (5.72 versus 6.86 reviews per retained item; first-pass 0.467 versus 0.471) but used a buggy Brier estimator and is reported here only as history. Because the shipped variant has no E multiplier, the ladder cannot move S' on a pass; it moves observed difficulty and effort. The fixture table below holds S, D, and elapsed fixed and shows exactly that: stability and interval are flat across five increasingly troubled passes, while D and effort climb monotonically.

Signal (failedRuns, hintTier)	phi	D'	effort'	Interval (days)
(0, 0)	0	4.7	0	32
(1, 0)	1	4.85	0.0625	32
(2, 1)	2.5	5.008824	0.15625	32
(3, 2)	4.5	5.15	0.28125	32
(2, 3)	6	5.225	0.375	32

Evidence reaches D and effort, not S (S = 10, D = 5, elapsed = 10, pass). In the shipped noE variant every signal produces the same stability S' = 32.026729 and interval 32; the ladder moves observed difficulty and effort. phi =

min(12, failedRuns + reset) + hintDebit[hintTier].

The difficulty channel compounds across reviews, which is the point. A clean pass and a pass with three failures and a tier-3 hint produce the same first-pass stability and interval, but different D (4.7 versus 5.265385); on the next clean pass at the scheduled interval the twin stabilities diverge to 93.041293 versus 87.565627, a ratio of 1.0625. Other fixtures pin the model down: R(S, S) = 0.9 to 1e-12 and I(0.9, S) = S to 1e-9 for S in {0.1, 1, 2.3065, 10, 100, 1000, 36500}; the cap fixture (S = 36500, D = 1, effort = 0.5, phi = 16) yields S' = 36500, D' = 2.063636, effort' = 0.75, and interval 7; one heavy four-failure lapse from effort = 0 gives effort' = 0.5 and is not capped, with S' = 3.747287 at S = 100 and interval 4. On ordering, an item with S = 100 due yesterday has R = 0.89931494 and an item with S = 3 due ten days ago has R = 0.77441687, so the second is served first. The original universal claim that the clean interval is at least 1.25x the struggled interval whenever phi >= 4.5 was refuted during verification (minimum 1.0; 975 counterexamples, e.g. S = 0.1, D = 1, rPred = 0.9, phi = 5 gives 1 versus 1), and is not claimed here.

5. Limitations & threats to validity

The most important limitation is that the truth model is synthetic and is itself built from FSRS-6. The simulation tests whether LGS schedules coherently against that model, not whether the model matches how humans forget multi-step code problems. Execution-verified, multi-minute problems occupy a measurement regime the flashcard benchmarks do not cover, so the reported percent-fewer-review figures are a calibration argument, not a claim about a classroom.

The hint debits have no published support. Nothing in the literature maps hint tiers or failed runs to a stability multiplier, and the hintDebit values used here are chosen, swept, and first to cut if calibration fails: the documented kill order is to lower the tier debits, then drop hints from phi entirely. Reset handling is the same kind of choice. The reset debit of 0.5 is a guess that a reset is sometimes refactoring rather than forgetting; the simulation's synthetic hint policy cannot validate that interpretation.

Dropping the E multiplier is a real trade. In noE, a single struggled pass schedules like a clean one, and the ladder bites only through D and through the effort cap over repeated reviews. If the truth model understates the cost of struggle, fragile items are over-scheduled. The measured edge cohorts argue the trade is safe under this protocol (hint-heavy 0.9933x, reset-heavy 0.7755x, one-leech 0.8375x SM-2 reviews), but the argument is only as good as the protocol.

The literal protocol is degenerate. Every arm retains 4000/4000 items, so literal numbers carry almost no signal, and the original spec's acceptance rule of 10% fewer reviews at equal first-attempt recall was satisfied there by arithmetic rather than by scheduling quality. Acceptance had to move to the memory-based truth model with ability offsets, which is a stronger but still synthetic comparison.

- No per-review log: personalization, FSRS optimization, and statistical leech detection are all out of reach in the current app.
- Fixed parameters: the benchmark shows optimized parameters beat defaults for most collections; LGS deliberately gives that up.
- The Elo-style difficulty update is weaker than a fitted IRT model and uses a single local user's outcomes.

- Leech thresholds remain practice lore; the effort cap is the weakest chosen default and is removed if it costs more than 3% reviews per retained item.
- Interleaving is unchanged and coarse: the confusability moderator that drives interleaving benefits (Brunmair & Richter 2019) is not modeled by the category-level interleave.
- Calendar-day elapsed time is DST-safe by construction but coarse; sub-day spacing is not modeled.
- The readiness change is user-visible: the 0.7 crossing moves from 8.4 days under the old linear curve to 21.42 days under the power law (R(21) = 0.70194, R(22) = 0.69741), and ordering changes on first read even though due dates do not.

Several theoretical claims from the first draft did not survive verification and are not repeated as results: the lapse fixtures were wrong until rPred was stated explicitly and the corrected values (0.316769 / 1.391987 / 3.747287, intervals 1 / 1 / 4) were adopted; the effort trajectory cannot be entered by a single clean pass because a clean pass has e = 0 and halves the accumulator; the old minimum pass ratio of 0.4 was corrected first to the true E floor of 0.40273 and then to 1.000900 once E was removed; and the 'migration runs on every read' formulation silently reset graded records until the guarded version was specified. These corrections are part of the record, because they mark where the design was fragile.

6. Reproducibility

Everything reported here is reproducible from the repository without network access, external data, or new dependencies. The scheduler is a pure TypeScript module, the simulation is a single script with a deterministic LCG, and the test suite covers the fixtures with fixed seeds.

- Core: src/lib/lgs.ts, pure and zero-import, with no Date.now, new Date, or Math.random inside.
- Simulation: scripts/lgs-sim.ts, seeded mulberry32 with fixed seeds 20260918 and 987654321; two consecutive runs must be byte-identical.
- Fixtures: tests/lgs.test.ts covers the identities, the ladder table, lapse values, the cap, the difficulty twin, ordering, migration, NaN guards, and elapsed-day rules.
- Migration fuzz: 3,000 fixed-seed payloads in the unit suite and 15,000 in the verifier run; a valid version-2 record must be unchanged by double migration, and due dates must never move without a grade.
- Determinism: the scheduler returns identical JSON for identical (state, signal, now) inputs, and identical {due, retrievability} for identical (state, now).

Acceptance is programmatic. The simulation exits non-zero unless matched-recall fewer reviews are at least 10%, Brier is no worse than SM-2 plus 0.01, every edge policy stays at or below 1.10x SM-2 reviews, and the two runs agree byte for byte. The single-seed ablation table and the 30-seed confidence intervals are both produced by that script; the 30-seed run freezes the seed list, so the interval is deterministic rather than resampled at report time. The gradient check is not part of CI: the script is manual, and the unit suite carries only fast fixtures.

Reproducing the reported results

```
# fixtures and migration fuzz
bun test tests/lgs.test.ts tests/inventions.test.ts
```

```
# deterministic simulation (manual; exits 1 if any gate fails)
bun run scripts/lgs-sim.ts

# byte-identity check for the two fixed seeds
bun run scripts/lgs-sim.ts > /tmp/lgs-run-1.txt
bun run scripts/lgs-sim.ts > /tmp/lgs-run-2.txt
cmp /tmp/lgs-run-1.txt /tmp/lgs-run-2.txt
```

The first-grade sequence from a seeded record is a useful smoke test of the whole pipeline: a clean first grade yields S' = 4.232585, D' = 4.7, and interval 4; grades 4 and 3 keep interval 4 with D' = 4.85 and 5.008824 and effort 0.0625 and 0.15625; grade 0 yields S' = 0.316769, interval 1, and D' = 5.75. The canonical later sequence is 4, 15, 49, 142, 375 days.

7. Future work

The clearest next step is an append-only per-review log. It is the prerequisite for everything the current design gives up: per-user FSRS weight optimization, statistical leech detection based on the Poisson-binomial probability of observed failures under predicted recall, and any serious attempt to calibrate hint debits against real outcomes. The benchmark's own caution applies in reverse: with a log and a few hundred reviews, personalization can start to beat the fixed defaults that LGS deliberately ships.

- Calibrate or remove the hint ladder: collect execution outcomes with hint tiers and failed runs, then fit a debit curve or cut the tier debits as the documented kill order prescribes.
- Validate the D and effort channels against real data; the synthetic truth model cannot rule out over-scheduling fragile items after a single hard pass.
- Couple with the out-of-scope hypotheses: horizon-scaled second intervals (H3) and deadline-aware caps (H5), both of which compose with predicted retrievability.
- Re-evaluate the removed E multiplier if a per-review log ever shows that struggled passes need a stability discount; the noE decision is an empirical result under this protocol, not a theorem.
- Adopt FSRS-7 fractional intervals once the fsrs-rs integration ships widely; its benchmark log loss of 0.3401 versus 0.3620 for defaults suggests headroom, though again on flashcard data.
- Explore objective execution features beyond counts: which test failed first, error class, and run duration, while keeping determinism and privacy intact.

> *Open questions — Does the ladder's difficulty signal remain calibrated when learners use the same hint tier for different reasons? Is the 0.5 reset debit meaningful outside refactoring? Does removing E over-schedule fragile items in a population with more struggle than the synthetic hint policy produces? None of these can be answered without real execution logs; they are listed so the design is judged against them rather than shielded from them.*

8. Conclusion

Ladder-Graded Spacing keeps the evidence that a code-practice platform already owns. Failed test runs, earned hint tiers, and resets become an effective failure count that updates observed difficulty and an effort EWMA, while scheduling stays on a fixed-parameter FSRS-6 power-law core at a 0.9 target. The ablation that removed the originally proposed stability multiplier made the design smaller

and measurably better: 25.63% fewer reviews than SM-2 at matched recall, a better Brier score on every seed, and edge cohorts at or below 1.10x baseline reviews. The remaining uncertainties are stated rather than hidden — synthetic truth, uncalibrated hint debits, and a ladder that in this variant moves difficulty and effort instead of stability. The technique ships with a guarded migration, deterministic simulation, and fixtures that make the next revision cheap to falsify.

References

1. Cepeda et al. 2006, distributed-practice meta-analysis (839 assessments). https://www.evullab.org/pdf/CepedaPashlerVulWixtedRohrer-PB-2006.pdf
2. Karpicke & Bauernschmidt 2011, absolute versus relative spacing. https://doi.org/10.1037/a0023436
3. Latimier et al. 2021, expanding versus uniform spacing meta-analysis. https://link.springer.com/article/10.1007/s10648-020-09572-8
4. Kang et al. 2014, expanding schedules on an eight-week test. https://doi.org/10.3758/s13423-014-0636-z
5. Storm et al. 2010, expanding schedules under interference. https://bjorklab.psych.ucla.edu/wp-content/uploads/sites/13/2016/07/Storm_Bjork_Storm_2010.pdf
6. Donovan & Radosevich 1999, spacing meta-analysis by task complexity. https://gwern.net/doc/psychology/spaced-repetition/1999-donovan.pdf
7. FSRS algorithm explainer (power-law curve, FSRS-6 weights). https://github.com/open-spaced-repetition/awesome-fsrs/wiki/The-Algorithm
8. open-spaced-repetition/srs-benchmark, calibration numbers and caveats. https://github.com/open-spaced-repetition/srs-benchmark
9. srs-benchmark pull request 290, FSRS-7 in the benchmark. https://github.com/open-spaced-repetition/srs-benchmark/pull/290
10. fsrs-rs pull request 395, FSRS-7 integration. https://github.com/open-spaced-repetition/fsrs-rs/pull/395
11. Expertium benchmark breakdown, log loss and RMSE(bins). https://expertium.github.io/Benchmark.html
12. DASH: Lindsey, Shroyer, Pashler & Mozer 2014, attempt-level frequency models. https://laplab.ucsd.edu/articles/LindseyShroyerPashlerMozer2014.pdf
13. DAS3H, attempt-level difficulty and skill model. https://arxiv.org/abs/1905.06873
14. HLR: half-life regression for recall prediction (ACL 2016). https://aclanthology.org/P16-1174/
15. Corbett & Anderson 1994, knowledge tracing. https://doi.org/10.1007/BF01099821
16. Piech et al. 2015, deep knowledge tracing. https://arxiv.org/abs/1506.05908
17. Mettler et al., ARTS: response time plus accuracy versus fixed spacing. https://pmc.ncbi.nlm.nih.gov/articles/PMC6028005/
18. SSP-MMC++: target-half-life and retention cost (Springer 2026). https://link.springer.com/chapter/10.1007/978-981-95-7072-0_47
19. Response-time propensities in step-level models (arXiv 2605.08943). https://arxiv.org/html/2605.08943
20. Kornell, Hays & Bjork 2009, unsuccessful retrieval with feedback. https://doi.org/10.1037/a0015729
21. Hays et al. 2013, failed retrieval and immediate feedback. https://doi.org/10.1037/a0028468
22. Stamper, LAK 2026, hint shortcuts and learning outcomes. https://dev.stamper.org/publications/An_LAK_2026.pdf
23. Maniktala 2022, hint use and help-need prediction. https://doi.org/10.1007/s11257-022-09338-7
24. Koedinger & Aleven 2007, the assistance dilemma. https://doi.org/10.1007/s10648-007-9049-0
25. Pelánek 2016, Elo-style approximations of IRT quality. https://doi.org/10.1016/j.compedu.2016.03.017

26. Ye et al., KDD 2022, the FSRS algorithm paper. https://dl.acm.org/doi/10.1145/3534678.3539081
27. Anki manual, leeches (practice, no controlled trial). https://docs.ankiweb.net/leeches.html
28. Anki pull request 3929, FSRS-6 shipping in Anki 25.07. https://github.com/ankitects/anki/pull/3929
29. ts-fsrs, browser-ready TypeScript FSRS implementation. https://github.com/open-spaced-repetition/ts-fsrs
30. Rowland 2014, retrieval-practice meta-analysis. https://doi.org/10.1037/a0037559
31. Bjork & Bjork 2020, desirable difficulties and their floor (JARMAC). https://sites.lifesci.ucla.edu/psych-bjorklab/wp-content/uploads/sites/13/2021/01/RABjorkELBjorkJARMAC2020ForPostingSingleSpaced.pdf
32. Brunmair & Richter 2019, interleaving meta-analysis. https://doi.org/10.1037/bul0000209