ATSOLVER: AGENTIC TUNING OF SPARSE LINEAR SOLVERS WITH NUMERICAL FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse linear solver configuration is a sequential decision problem under numerical, resource, and workload constraints. Each attempted configuration reveals execution evidence that can inform subsequent choices, while setup cost, solve cost, accuracy requirements, and search budgets jointly shape the tuning objective. We introduce **ATSOLVER (Agentic Tuning of Solvers)**, a novel verifier-grounded framework for agentic sparse-solver tuning that iteratively proposes, executes, and revises configurations over a typed portfolio of classical sparse solvers and preconditioners. A pretrained language model with fixed parameters serves as the proposal policy, while trusted numerical software controls legality, execution, solution verification, resource accounting, and numerical-state reuse. Under matched six-proposal budgets in an iterative-solver study, ATSOLVER accepts 75/105 runs versus 64/105 for independent Best-of-6 and selects configurations with 30.1% lower independently replayed solve time on matrices solved by both policies in all repetitions. Across repeated-RHS workloads, ATSOLVER shifts toward greater setup investment as batch size grows, adapting configuration choices to changing setup–solve trade-offs. Feedback memory studies show that trial history avoids observed repeated proposals and that adding retrieved experience to knowledge improves configuration quality on held-out parameters within two PDE families. These results demonstrate effective agentic tuning under explicit accuracy, runtime, and memory constraints without updating the underlying language model.

1 INTRODUCTION

Sparse linear systems are central to simulation, optimization, and scientific machine learning, yet their performance can vary substantially with the solver, preconditioner, and numerical parameters used. Choosing these configurations requires balancing convergence, setup cost, solve cost, and memory use (Demmel et al., 2005; Pearson & Pestana, 2020). The preferred choice also depends on the workload: an expensive setup may be unattractive for a single solve but economical when reused across many right-hand sides (RHSs). Moreover, several quantities that matter for configuration—including convergence behavior, achieved residual, runtime, and memory use—are revealed only after a candidate is executed. Solver configuration therefore benefits from not only problem information available before execution, but also evidence accumulated during the search.

Existing approaches exploit different parts of this information. Learned solver selectors use matrix and performance features to recommend configurations (Jessup et al., 2016; Tang et al., 2022), while autotuning methods use measured outcomes to guide search (Hutter et al., 2011; Ansel et al., 2014). More recently, language-model optimization has shown that pretrained models can generate and refine proposals using previous evaluations (Yang et al., 2024; Liu et al., 2024). However, sparse-solver configuration poses a particularly structured version of this problem. A useful method must reason over solver and preconditioner choices, numerical parameters, resource limits, and setup–solve trade-offs, while ensuring that every accepted solution satisfies the original numerical problem. This motivates a feedback-driven configuration process in which proposals can be revised after execution without giving the language model authority over numerical correctness.

To this end, we introduce ATSOLVER (Agentic Tuning of Solvers), a verifier-grounded framework for sparse-solver configuration. At its core, ATSOLVER is an iterative configuration-refinement method that proposes, executes, evaluates, and revises classical solver configurations using numeri-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

cal feedback. Specifically, a pretrained language model proposes a solver–preconditioner configuration, numerical parameters, and lifecycle choices through a typed interface. Numerical software then validates and executes the proposal, independently verifies the resulting solution, and returns the observed outcome for subsequent decisions. Failed trials can guide parameter changes or switches to other solver branches, while successful trials can be further refined toward lower deployment cost. The best verified configuration is retained throughout the search. For repeated RHSs, ATSOLVER can additionally trade greater setup investment against lower per-RHS solve cost. The same language model is used throughout, with adaptation occurring through execution feedback rather than parameter updates.

We evaluate ATSOLVER on 35 sparse matrices spanning 32 families and multiple workload settings. Under matched six-proposal budgets in an iterative-solver study, ATSOLVER accepts 75/105 runs compared with 64/105 for independent Best-of-6, while its selected configurations achieve 30.1% lower independently replayed solve time on matrices solved by both methods in all repetitions. In a separate three-proposal study that also includes direct solvers, ATSOLVER accepts 78/105 runs versus 74/105 for Direct Best-of-3 and reduces deployment cost by 17.2% relative to that baseline on jointly accepted runs. Across repeated-RHS workloads, ATSOLVER increasingly selects configurations with larger setup investment as the batch size grows, reflecting the changing setup–solve trade-off. Controlled experiments further examine how trial history, retrieved experience, and matrix information contribute to this iterative configuration process.

Contributions. We make three contributions:

1. **Workload-aware formulation of sparse-solver configuration.** We formulate solver configuration as a workload-conditioned sequential decision problem under explicit accuracy, resource, and search budgets, separating the cost of finding a configuration from the cost of deploying it across single- and multi-RHS workloads.
2. **Verifier-grounded agentic tuning.** We introduce ATSOLVER, a sequential controller that revises classical solver configurations using execution evidence, while numerical software retains authority over proposal validity, execution, independent verification, and numerical state.
3. **Controlled Studies for execution feedback and matrix images.** Through matched-budget comparisons and controlled studies, we show that trial history avoids redundant proposals, retrieved experience improves configuration quality on held-out PDE parameters, and larger RHS workloads shift configuration choices toward greater setup investment and lower per-RHS solve cost. A separate image ablation finds lower replayed solve time with structural views.

2 WORKLOAD AND CONFIGURATION OBJECTIVE

We consider three workload regimes. Regime S contains a single linear system, regime M contains multiple right-hand sides (RHSs) sharing the same matrix, and regime Q contains a sequence of related systems in which the operator may change between solves. We mainly develop S and M; regime Q is evaluated as an extension. For regimes S and M, we consider K systems $Ax_k = b_k$, $k = 1, \dots, K$, with sparse $A \in \mathbb{R}^{n\times n}$ and RHSs $b_k \in \mathbb{R}^n$. We collect the RHSs as $\mathbf{B}_{\mathrm{rhs}} = [b_1, \dots, b_K] \in \mathbb{R}^{n\times K}$. Regime S has $K = 1$, while M has $K > 1$ with the same matrix used for every solve. We write the workload contract as $\mathcal{W} = (L, R, H, B)$, where L specifies the workload regime and linear-system characteristics, R the accuracy and coverage requirements, H the available hardware and solver capabilities, and B the resource and search budgets. A configuration a specifies a solver, preconditioner, numerical parameters, and permitted preprocessing or setup-reuse operations. Within the search budget, we first seek a configuration that satisfies the complete workload, then prefer lower deployment cost among accepted candidates.

Numerical acceptance. Numerical software independently verifies each computed solution $\widehat{x}_k$ by recomputing its residual on the original system:

$$\rho_k = \frac{\|A\widehat{x}_k - b_k\|_2}{\max\{\|b_k\|_2, \epsilon_b\}}, \tag{1}$$

where $\epsilon_b > 0$ safeguards zero or near-zero RHSs. Every required RHS must satisfy its prescribed tolerance $\rho_k \le \tau_k$. Acceptance also requires finite outputs, method validity, and compliance with time and memory limits.

Search limits and cost. The controller may submit at most B_{prop} proposals and execute at most B_{exec} configurations. Every submission counts toward the proposal limit, including invalid, repeated, or unaffordable configurations and premature STOP requests. Only proposals that pass software validation are executed and count toward the execution limit. Each execution has a timeout, and the workload has an overall deadline; treatment of model latency is specified in advance. Let $g_j = 1$ if proposal j is executed and $g_j = 0$ otherwise. For an executed trial, C_j^{num} includes preprocessing, setup, solves, verification, and permitted state-transfer work. Total numerical search cost is

$$C_{\text{ctrl}} = C_{\text{card}} + \sum\nolimits_{j:g_j=1} C_j^{\text{num}}, \tag{2}$$

where C_{card} is the one-time numerical cost of preparing the workload summary. Successful and failed executions both contribute. Work to first acceptance, C_{first}, includes C_{card} and all executed trials through the first accepted attempt, and is infinite if none is accepted. Elapsed search time also includes model and orchestration latency; token use and provider charges are reported separately.

Deployment cost. For regimes S and M, deployment cost is the time required to run the selected configuration on the workload, excluding the search used to find it:

$$C_{\text{deploy}}(a; K) = C_{\text{pre}}(a) + C_{\text{setup}}(a) + \sum_{k=1}^{K} C_{\text{solve},k}(a) + C_{\text{transition}}(a). \tag{3}$$

The terms measure preprocessing, setup, RHS solves, and additional state-transfer work. Verification time is reported separately. When setup is reused across RHSs, greater initial investment may be offset by lower subsequent solve cost; the multi-RHS study tests whether configuration choices reflect this trade-off. Regime Q extends this accounting to related-system sequences and permitted cross-system state transfer, as described in Appendix D.7. Each study distinguishes in-search timing from independent replay and measured batch costs from projections based on single-RHS timings.

3 ATSOLVER

Decision inputs and actions. Before proposal j, the model receives workload information c_j, the permitted trial history m_j^{arm}, legal actions $\mathcal{A}_j$, and remaining budgets β_j. Together, these form the decision state d_j from which the policy π_θ generates its next action:

$$d_j = (c_j, m_j^{\text{arm}}, \mathcal{A}_j, \beta_j), \qquad a_j \sim \pi_\theta(\cdot \mid d_j). \tag{4}$$

The workload information includes matrix dimensions, nonzero counts, structural and scaling properties, available numerical estimates, accuracy and resource requirements, and solver capabilities. Where enabled, matrix views and retrieved records provide additional context. Matrix names, file paths, private numerical objects, and private verifier state are withheld. Appendix C.1 describes the optional model inputs, and the numerical cost of constructing the workload summary is counted once. Each numerical action selects a solver and preconditioner, sets their parameters, and specifies permitted preprocessing, initial-guess, and setup operations. The portfolio includes PCG, GMRES, FGMRES, and BiCGSTAB with supported AMG, ILU, FSAI, or no-preconditioner options, plus serial SuperLU where permitted. Proposals use a typed interface restricted to implemented workflows; the model does not generate solver code. Appendix C describes the interface, and Table 10 lists its parameters.

Execution evidence and revision. An executed trial returns its acceptance status, residuals, available phase timings, and memory measurements; a proposal rejected before execution instead returns a rejection reason. Each observation remains linked to its configuration. Rejected proposals are not automatically repaired, and unavailable measurements remain missing. The permitted execution evidence can condition the next configuration decision. The model may revise parameters within the current solver–preconditioner combination or switch to another combination. After a failure, a deterministic diagnostic procedure summarizes the observed failure type and identifies aspects that may need revision; it does not select the next solver. After success, setup and solve measurements can guide further proposals toward lower deployment cost. The resulting history and remaining budgets form the next decision state, while the experimental condition determines which portions of the recorded history and feedback are exposed to the model. Appendix C.4 specifies these projections.

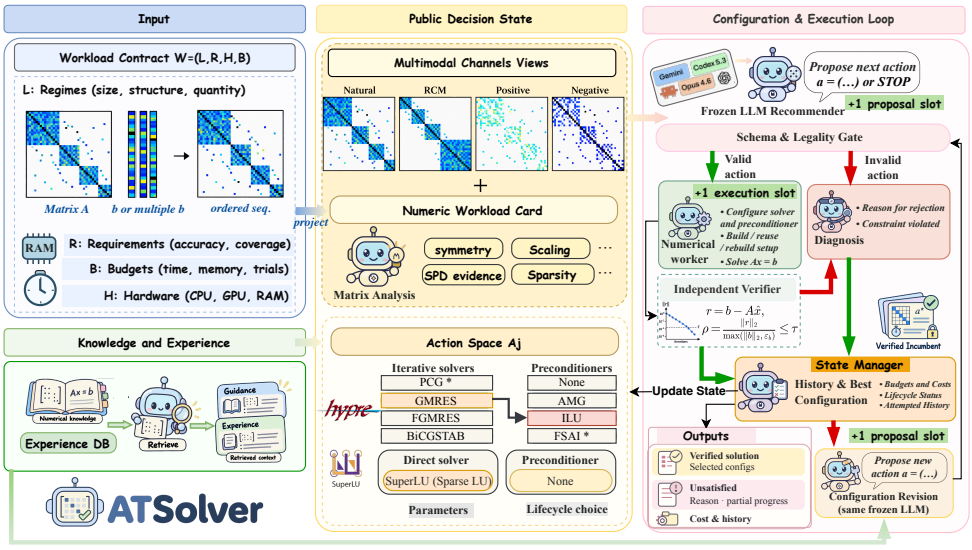


Figure 1: **Verifier-grounded configuration in ATSOLVER.** Workload information, optional matrix views, and retrieved records inform configuration proposals. Recommendation and revision use the same pretrained language model with fixed parameters. Numerical software controls proposal validation, execution, solution verification, budget accounting, and permitted state reuse. The state manager retains the best accepted configuration and updates the evidence available to subsequent proposals. Optional inputs are enabled only in the specified experimental conditions.

Best accepted configuration and numerical state. Software retains the best accepted configuration, replacing it only when another accepted candidate improves on it under the fixed selection rule. Later failures leave it unchanged. Single-system search continues within the fixed proposal horizon after first acceptance to seek further improvements. Trial history and numerical state are distinct. The model may use earlier trial information, but single-system and multi-RHS candidates do not inherit setup objects or solutions from other trials. Each single-system candidate starts from the same prescribed numerical state, without continuing an earlier iterate. Each multi-RHS candidate must solve the entire batch; successful solves from different candidates cannot be combined. Within one multi-RHS candidate, however, a setup may be built once and reused across RHSs. Batch size therefore informs whether a larger setup investment is worthwhile. An earlier RHS solution may also serve as an initial guess when the workload and backend permit it. For related-system sequences, only the selected accepted candidate may pass validated numerical state to the next system. Appendix C.5 gives the checks required when the matrix changes.

4 EXPERIMENTAL SETUP

Data and implementation. The configuration-quality, within-search history, and batch studies use 35 sparse matrices: 29 from SuiteSparse and six assembled with MFEM, spanning 32 families. Three model repetitions per workload give 105 runs per condition. Separate development data support baseline construction and implementation checks. Related matrices remain in the same partition, and matrix names are withheld from the model. We use `gpt-5.6-terra`, HYPRE 3.1, and serial SuperLU 7. Appendices B.1 and B.2 give the data and implementation details.

Configuration quality (RQ1). We compare ATSOLVER with Direct and Expert-guided Best-of-3, which omit explicit trial histories but retain the shared diagnostic handoff after failures. Expert-guided receives the same fixed solver-selection advice as ATSOLVER; Direct omits it. All three share the numerical portfolio, three-proposal and three-execution limits, and selection rule. Direct refers to the prompt, not the numerical solver type. One-shot variants and fixed GMRES+AMG provide single-configuration references. Also, a separate iterative-only study gives ATSOLVER and independent Best-of-6 the same matrix images, retrieved knowledge, and six proposal and execution slots. Selected configurations receive one warmup and five measured replays with fresh setup and

verification. Replayed solve time is the median of the five measured solve times. These results are reported separately from the three-proposal study.

Values of Feedbacks (RQ2). We examine feedback memory within and across configuration searches. Within a search, we compare no action history, tried configurations alone, and configurations with scalar outcomes or structured diagnostics under matched proposal and execution limits. All conditions retain the shared diagnostic handoff. We assess whether explicit history reduces duplicate proposals and improves numerical acceptance.

Across workloads, we compare knowledge alone (K) with the same knowledge plus retrieved trial experience (K+E), holding online feedback, other inputs, and execution settings fixed. We use held-out parameters within the Helmholtz and SUPG families to test whether prior experience improves performance-target attainment during search and the solve performance of selected configurations under independent replay. Appendices D.2 and D.6 provide the detailed comparisons.

Setup investment for repeated solves (RQ3). We vary batch size over $K \in \{2, 8, 32\}$, requiring each candidate to solve every RHS. We compare ATSOLVER with an expert rule using setup–solve cost estimates, a trained configuration ranker, and model-based search that updates predictions from trial outcomes. All four methods have three proposal and execution slots. The controls choose among five preset configurations, whereas ATSOLVER also tunes parameters. Thus, trial limits match but available choices differ. Appendices B.3 and B.4 specify the controls and batch protocol.

Matrix views. On eight matrices, the image study holds numerical summaries and retrieved knowledge fixed while comparing no images, two structural views, and four combined structural and signed-value views. The study uses iterative solvers, six proposals, and three repetitions, giving 24 runs per condition. Other inputs and execution settings remain fixed (Appendices C.1 and B.4).

Evaluation criteria. We report counts of numerically accepted runs (*Accepted*), including cumulative counts within q proposals in the within-search history study. For retrieved experience, online target attainment counts searches with any qualifying trial within q proposals; final replay attainment evaluates the selected configuration using its median replayed solve time. Both require numerical acceptance and the specified solve-time target. Failed searches remain included, and rejected proposals count toward the budget. Every required system must pass for a batch or sequence to succeed. Acceptance gains use percentage points; duplicate rates use all proposals as the denominator.

1st-Accept Work is the restricted mean numerical work to first acceptance over a 180 s horizon in the three-proposal single-system study, retaining unsuccessful searches. Full search time includes model and orchestration latency. Solve time measures the solve phase; deployment includes preprocessing, setup, the required solves, and any prescribed state-transfer work. Both exclude verification and search. Paired timings use the stated jointly accepted subsets, with run and matrix counts reported. A tested-condition/reference time ratio r corresponds to a $100(1 - r)\%$ reduction.

5 RESULTS

We evaluate configuration quality and search cost (RQ1), the contributions of within-search history and retrieved experience (RQ2), and setup investment across RHS batch sizes (RQ3), followed by matrix-view ablations. Each study follows the workloads, budgets, and timing protocol in Section 4.

5.1 RQ1: CONFIGURATION QUALITY AND SEARCH COST

Closed-loop configuration search. ATSOLVER uses accumulated trial feedback to revise configurations, beyond the shared diagnostic handoff available to the Best-of-3 controls. It finds accepted configurations more often than the one-shot and fixed GMRES+AMG references, reaching 78/105 runs versus 74/105 for Direct Best-of-3 and 75/105 for Expert-guided Best-of-3 under the matched three-proposal budget (Table 1). Among the matched-budget methods, it requires the least numerical work to first acceptance: a restricted mean of 48.76 s versus 54.06 s for Expert-guided Best-of-3. This comes with longer median full search time: 26.28 s versus 19.90 s.

Table 1: **Matched-proposal configuration quality and search cost.**

(a) Acceptance and search

Method	Accepted ↑ ($n/105$)	1st-Accept Work ↓ (s; restricted mean)	Full search ↓ (s; median)
Single-configuration references ($1/1$ *limits*)			
Fixed GMRES+AMG	51	93.11	1.92
Direct one-shot	70	61.37	8.14
Expert-guided one-shot	65	69.85	8.04
Matched three-proposal searches ($3/3$ *limits*)			
Direct Best-of-3	74	55.28	21.64
Expert-guided Best-of-3	75	54.06	**19.90**
ATSOLVER	**78**	**48.76**	26.28

(b) Paired deployment reduction

Reference	Paired runs	Matrices	Deploy reduction ↑
Direct Best-of-3	74	26	**17.2%**
Expert-guided Best-of-3	75	25	**22.0%**

Deployment uses in-search timings, geometrically aggregating ratios of within-matrix method medians. Throughout Tables 1–4, bold marks best comparable values or positive gains, not statistical significance.

Cheaper workflows beyond faster solves. On each comparison's jointly accepted runs, ATSOLVER reduces deployment time by 17.2% relative to Direct Best-of-3 and 22.0% relative to Expert-guided Best-of-3 (Table 1(b)). These savings include preprocessing, setup, and solve, demonstrating gains in complete numerical workflows. Whether repeated deployment amortizes the additional search cost depends on workload reuse.

In the separate six-proposal, iterative-only study, ATSOLVER accepts 75/105 runs versus 64/105 for independent Best-of-6 (Table 2). Independent replay yields 30.1% lower solve time and 4.5% lower deployment time on the paired subset; full search time is 75.23 s versus 53.37 s.

Table 2: **Six-proposal iterative study.** Replay ratios use 60 paired runs on 20 matrices accepted by both policies in all three repetitions.

Method	Accepted ↑ ($n/105$)	Solve ↓ ratio	Deploy ↓ ratio	Full search ↓ (s)
Independent Best-of-6	64	1.000	1.000	**53.37**
ATSOLVER	**75**	**0.699**	**0.955**	75.23

Ratios are ATSOLVER/Best-of-6. Full search is the median of within-matrix repetition medians over all 35 matrices, including failures.

From recovery to refinement. On `bcsstk17`, the initial PCG+AMG configuration reaches its iteration limit before meeting the requested accuracy. Raising the limit produces an accepted solution; increasing AMG sweeps then produces another accepted configuration with lower recorded deployment cost (Appendix E.1, Table 24). Each trial starts from fresh numerical state rather than continuing an earlier iterate. Across the three-proposal single-system study, ATSOLVER recovers 18 of 45 initially unsuccessful searches on their second proposal. A separate, larger-budget seeded-continuation study also refines already accepted seed configurations; its outcomes and search costs appear in Appendix D.1.1.

5.2 RQ2: DOES MEMORY IMPROVE CONFIGURATION SEARCH?

Within-search memory avoids repeated proposals. Under a three-proposal horizon, no duplicate proposals are observed in any condition that exposes tried configurations, compared with 94/315 (29.8%) without explicit action history. Since the gate charges rejected duplicates to the proposal budget, this memory avoids spending slots on previously tried configurations. Final acceptance is 77/105 without action history and 78/105 in each history condition. Appendix D.2 reports the full history comparison and the constructed-failure intervention, where structured diagnostics yield fewer accepted next actions than scalar outcomes (62/105 versus 68/105).

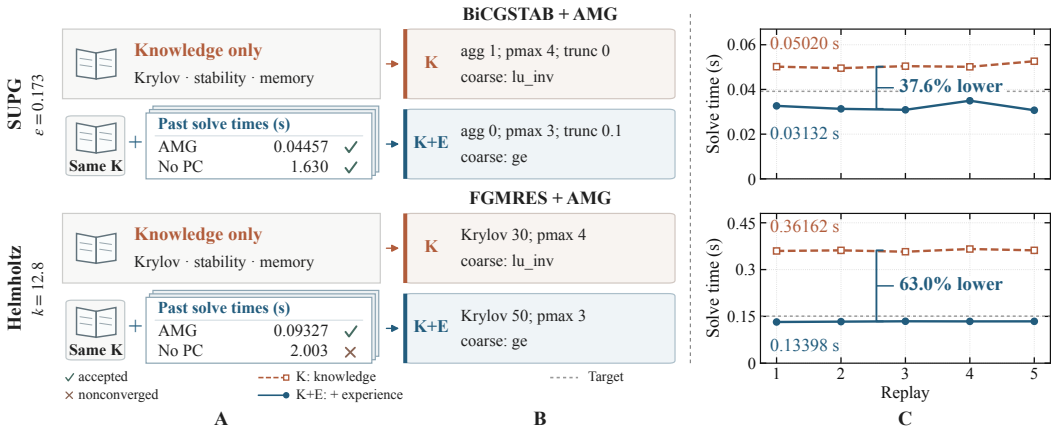


Figure 2: **Selected knowledge–experience cases.** (A) Knowledge and retrieved trial records. (B) Selected parameters. (C) Five replays with knowledge alone (K) and added experience (K+E). Reductions are case-level median comparisons; dotted lines mark the performance target. The text reports study-wide aggregate effects.

Retrieved experience improves configuration quality. The comparison in Table 3 tests retrieved memory with knowledge and online feedback present in both conditions. With experience (K+E), 88.1% of searches reach the performance target after one proposal and 100% within three, compared with 31.0% and 42.9% for knowledge alone (K). K reaches 54.8% within six proposals. These online rates count any proposal reaching the target. Independent replay instead evaluates the final selected configurations, giving 38/42 (90.5%) versus 19/42 (45.2%). Both conditions satisfy all 42 numerical contracts; the gain concerns solve performance rather than additional numerical acceptance.

Adding experience reduces paired replayed solve time by 11.0%, deployment time by 3.8%, and numerical search cost by 44.5% (Appendix D.6). Both conditions use all six calls; experience increases tokens by 36.2% and modeled API charges by 31.4%. Figure 2 illustrates selected configuration changes and replayed timings. These results support using prior trial outcomes alongside knowledge and online feedback to improve configuration quality on held-out parameters within the two PDE families.

Table 3: **Retrieved experience and performance-target attainment.** Each condition has 42 six-proposal searches on 14 held-out parameter points.

Evidence	**Online attainment within q proposals (%) ↑**						**Final replay**
	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$	$q = 6$	(%) ↑
K	31.0	40.5	42.9	47.6	47.6	54.8	45.2
K+E	**88.1**	**95.2**	**100.0**	**100.0**	**100.0**	**100.0**	**90.5**
Δ (pp)	**+57.1**	**+54.8**	**+57.1**	**+52.4**	**+52.4**	**+45.2**	**+45.2**

The target requires numerical acceptance and solve time within $1.05\times$ the best stable fixed recipe; final attainment uses five-replay medians. Δ: K+E minus K in percentage points, computed before rounding.

5.3 RQ3: How do setup choices change with batch size?

Batch acceptance and deployment. ATSolver accepts 78, 75, and 72 of 105 batches at $K = 2, 8, 32$, respectively (Table 4(a)). At $K = 32$, the expert amortization rule accepts 50/105 batches, while model-based search matches ATSolver at 72/105. On the 50 jointly accepted runs across 17 matrices, configurations selected by ATSolver reduce deployment time by 48.5% relative to the expert rule. This comparison measures the cost of completing the entire RHS batch, including preprocessing, setup, and all required solves. Table 17 reports the remaining paired comparisons.

Greater setup investment with more RHS reuse. To track configuration choices on the same cases, Table 4(b) uses the 72 ATSolver matrix–repetition pairs accepted at every batch size. As K increases from 2 to 32, direct-factorization selections rise from 22/72 to 55/72. Median setup time increases from 0.0338 to 0.2152 s, while median solve time per RHS falls from 0.03595 to 0.00772 s. Because this comparison holds the evaluated cases fixed, the shift is not explained by changes in the accepted subset. The observed choices are consistent with spreading a more expensive setup over

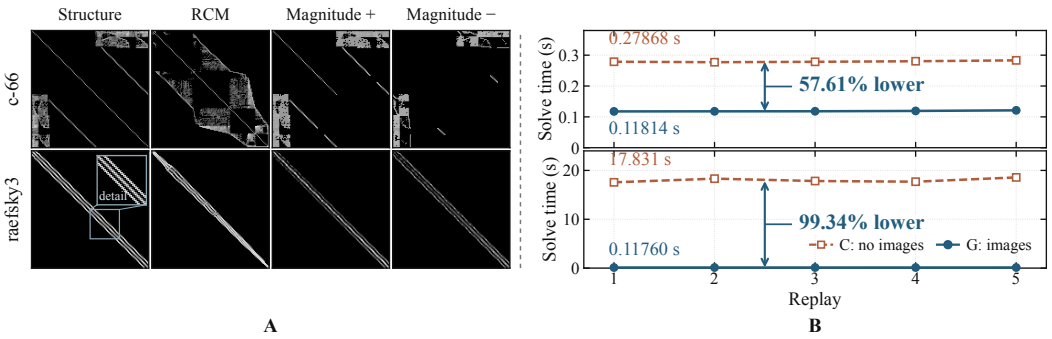


Figure 3: **Selected matrix-image cases.** (A) Four-view inputs. (B) Five timing replays with four views (G) and without images (C). Reductions compare medians for these selected cases; the text reports aggregate effects across the study.

more RHSs to reduce the cost of subsequent solves. ATSOLVER has higher median full search time than the expert rule at every K (Table 4(c)); deployment savings exclude this overhead.

Related-system sequences (Q). Joint control accepts 72/105 four-position sequences under each schedule, versus 63/105 for carryover with prescribed rebuilding when the operator changes at every transition and 69/105 under mixed changed and unchanged transitions (Table 4(d)).

These batch results show that ATSOLVER adapts setup–solve trade-offs to RHS reuse, with its largest observed deployment gain against the expert rule at $K = 32$.

Table 4: **Complete-workload acceptance and paired deployment gains.**

(a) M: RHS batches, $3/3$ proposal/execution limits

Method / reference	$K = 2$		$K = 8$		$K = 32$	
	Acc.↑	**Deploy red. (%)**	**Acc.↑**	**Deploy red. (%)**	**Acc.↑**	**Deploy red. (%)**
Expert amortization rule	69/105	**7.1**	66/105	**11.2**	50/105	**48.5**
Supervised ranker	69/105	**6.1**	66/105	**3.2**	57/105	**2.0**
Model-based search	75/105	**5.5**	69/105	−1.7	**72/105**	**2.2**
ATSOLVER	**78/105**	–	**75/105**	–	**72/105**	–

(b) Configurations selected by ATSOLVER

Same 72 matrix–repetition pairs accepted at every K (24 matrices).

Quantity	$K = 2$	$K = 8$	$K = 32$
Direct factorization selected	22/72	41/72	55/72
Median setup time (s)	0.0338	0.0945	0.2152
Median solve time per RHS (s)	0.03595	0.01198	0.00772

(c) Full search time: all 105 searches per method

Method (median seconds)	$K = 2$	$K = 8$	$K = 32$
ATSOLVER	32.68	42.24	65.46
Expert amortization rule	**9.53**	**21.68**	**62.55**

(d) Q: four-position sequences, $7/7$ proposal/execution limits

Policy	**All changed: accepted ↑**	**Mixed: accepted ↑**
Carryover + prescribed rebuild	63/105	69/105
ATSOLVER (joint structured)	**72/105**	**72/105**

Deployment reductions use in-search ATSOLVER/reference ratios, geometrically aggregated within and across matrices on jointly accepted batches (Table 17).

5.4 MATRIX VIEWS

This separate iterative-only study uses six proposals and three repetitions per case, holding other inputs and execution settings fixed. The full table appears in Appendix D.5.

Multimodal matrix views. All three image conditions accept 18/24 runs on eight matrices. On the six jointly successful matrices, two sparsity views in the original and reverse Cuthill–McKee orderings reduce replayed solve time by 14.8% relative to no images; four structural and signed-value views give a smaller 9.6% reduction. Both image conditions increase online search time. Figure 3 shows selected cases alongside the four-view inputs.

6 Related work

Solver selection and configuration. Algorithm selection and configuration formalize instance-conditioned prediction and budgeted search (Rice, 1976; Xu et al., 2008; Schede et al., 2022). Self-adapting numerical software, Lighthouse, learned performance models, and matrix-based selectors recommend configurations from problem features and measured performance (Demmel et al., 2005; Jessup et al., 2016; Yeom et al., 2016; Tang et al., 2022). SMAC and OpenTuner provide general search procedures; Xabclib incorporates accuracy and resource use into runtime policies (Hutter et al., 2011; Ansel et al., 2014; Katagiri et al., 2024). We evaluate a pretrained proposal policy over a fixed classical portfolio, including configuration cost in the comparison.

Language-model-guided search. ReAct and Reflexion guide model actions with interaction histories; OPRO and LLAMBO condition optimization proposals on previous evaluations (Yao et al., 2022; Shinn et al., 2023; Yang et al., 2024; Liu et al., 2024). ATSolver exposes numerical measurements and diagnostics for revising sparse-solver configurations, with acceptance controlled by numerical software. Related systems generate code or broader PDE strategies (Cheng et al., 2026; Dong et al., 2026; Zhang et al., 2026); our controller configures existing solvers after matrix assembly. Complete-search comparisons evaluate the policy, while the common-failure experiment isolates the information supplied for the next decision.

Repeated solves and numerical verification. Block methods, warm starts, preconditioner reuse, and Krylov recycling reuse numerical work (O'Leary, 1980; Parks et al., 2006; Soodhalter et al., 2020). Our multi-RHS study examines selection among these existing setup–solve tradeoffs as batch size changes. Independent residual checks, motivated by reliable stopping criteria and backward-error analysis (Arioli et al., 1989; 1992), provide a common acceptance test. Appendix G expands on online configuration, experience retrieval, and benchmark resources.

7 Limitations and broader impact

Our findings concern the evaluated portfolio, 35 matrices, and the recorded models and hardware. Withholding matrix names does not rule out pretraining exposure, and residual acceptance does not certify forward error or physical-model accuracy. Several comparisons are deliberately scoped: the fixed-parent result concerns truncated GMRES trials, the batch controls select from a smaller configuration space than ATSolver, timing protocols and successful subsets differ across studies, and search costs can outweigh deployment savings. Retrieval depends on its corpus and split, and the parameter holdout is not family-disjoint. On broader impact, verifier-gated configuration can lower the expertise barrier for using classical solvers well, but it may also invite over-reliance on automated choices; independent verification and complete failure accounting limit automation bias, and users remain responsible for choosing appropriate numerical and resource requirements.

8 Conclusion

ATSolver pairs a frozen language-model proposal policy with verified execution of classical sparse solvers: the model chooses what to try, and software decides what counts. Under matched budgets, this closed loop finds acceptable configurations more often than independent sampling and selects workflows that are 17–22% cheaper to deploy, with 30.1% faster replayed solves in the iterative study; controlled ablations attribute the gains to acting on trial history rather than to diagnostic detail. Given repeated solves, the controller invests in setup where amortization justifies it, most visibly at 32 RHSs. These gains must still be weighed against search overhead, which we account separately and which is not always repaid.

AI USE STATEMENT

In this work, generative AI tools were used in two roles. First, language models were accessed through API calls as components of the evaluated ATSOLVER system, where they generated and revised solver-configuration proposals from the prescribed model-visible inputs. Second, generative AI tools were used to assist with manuscript drafting, editing, organization, literature search, and summarization. We reviewed all AI-assisted work and verified reported experimental results, scientific claims, citations, and artifacts against the underlying records and source materials. The authors take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

ATSOLVER automates solver configuration but does not replace numerical verification. A potential risk is over-reliance on automated configuration in scientific workflows; users remain responsible for specifying appropriate numerical and resource requirements and for interpreting the resulting solutions. The work does not involve human subjects or personal data.

REPRODUCIBILITY STATEMENT

Sections 2 and 3 specify the workload objective and ATSOLVER controller. Appendix B.1 documents the evaluation data and workload construction, and Appendix B.2 records the model, numerical software, and execution environment. Appendices C.1, C, C.4, and C.5 specify the model inputs, typed action interface, feedback projections, and numerical-state rules. The supplementary materials record per-run proposals, validity decisions, numerical executions, verification outcomes, budgets, API usage, and selected configurations, along with the procedures used to reconstruct the reported tables and figures.

REFERENCES

Patrick R Amestoy, Jean-Yves L'Excellent, and Gilles Moreau. On exploiting sparsity of multiple right-hand sides in sparse direct solvers. *SIAM Journal on Scientific Computing*, 41(1):A269–A291, 2019.

Jason Ansel, Shoaib Kamil, Kalyan Veeramachaneni, Jonathan Ragan-Kelley, Jeffrey Bosboom, Una-May O'Reilly, and Saman Amarasinghe. Opentuner: An extensible framework for program autotuning. In *Proceedings of the 23rd international conference on Parallel architectures and compilation*, pp. 303–316, 2014.

Mario Arioli, James W Demmel, and Iain S Duff. Solving sparse linear systems with sparse backward error. *SIAM Journal on Matrix Analysis and Applications*, 10(2):165–190, 1989.

Mario Arioli, Iain Duff, and Daniel Ruiz. Stopping criteria for iterative solvers. *SIAM Journal on Matrix Analysis and Applications*, 13(1):138–144, 1992.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

Arielle Carr, Eric de Sturler, and Serkan Gugercin. Preconditioning parametrized linear systems. *SIAM Journal on Scientific Computing*, 43(3):A2242–A2267, 2021.

Ao Cheng, Lei Zhang, and Guowei He. Re4: Scientific computing agent with rewriting, resolution, review and revision, 2026. URL https://arxiv.org/abs/2508.20729. Presented at the ICLR 2026 Workshop on AI and Partial Differential Equations.

Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *Acm transactions on mathematical software (toms)*, 38(1):1–25, 2011.

Jim Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Rich Vuduc, R Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312, 2005.

Huanshuo Dong, Keyao Zhang, Hong Wang, Zhezheng Hao, Zhiwei Zhuang, Ziyan Liu, Jiacong Wang, Gengyuan Liu, and Xin Jin. Autopde: Reliable agentic pde solving via explicitly represented solver strategies. *arXiv preprint arXiv:2606.10752*, 2026.

Paul R Eller, Jing-Ru C Cheng, and Robert S Maier. Dynamic linear solver selection for transient simulations using multi-label classifiers. *Procedia Computer Science*, 9:1523–1532, 2012.

Siyuan Guo, Yali Du, Hechang Chen, Yi Chang, and Jun Wang. Cascade: Case-based continual adaptation for large language models during deployment. *arXiv preprint arXiv:2605.06702*, 2026.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer, 2011.

Elizabeth Jessup, Pate Motter, Boyana Norris, and Kanika Sood. Performance-based numerical solver selection in the lighthouse framework. *SIAM Journal on Scientific Computing*, 38(5): S750–S771, 2016.

Takahiro Katagiri, Takao Sakurai, Mitsuyoshi Igai, Shoji Itoh, Satoshi Ohshima, Hisayasu Kuroda, Ken Naono, and Kengo Nakajima. Xabclib: A fully auto-tuned sparse iterative solver. *arXiv preprint arXiv:2405.01599*, 2024.

Mikhail Khodak, Edmond Chow, Nina Balcan, and Ameet Talwalkar. Learning to relax: Setting solver parameters across a sequence of linear system instances. In *International Conference on Learning Representations*, volume 2024, pp. 33688–33720, 2024.

Mikhail Khodak, Min Ki Jung, Brian Wynne, Edmond Chow, and Egemen Kolemen. One-shot acceleration of transient pde solvers via online-learned preconditioners. *Computer Physics Communications*, pp. 110304, 2026.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2402.03921. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/84b8d9fcb4e262fcd429544697e1e720-Abstract-Conference.html.

Hayden Liu Weng, Hans-Joachim Bungartz, and Felix Dietrich. Embedding-based methods for linear solver performance prediction. In *Computational Science – ICCS 2026*, volume 16783 of *Lecture Notes in Computer Science*, pp. 472–486. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-032-29921-5_32. URL https://doi.org/10.1007/978-3-032-29921-5_32.

Dianne P O'Leary. The block conjugate gradient algorithm and related methods. *Linear algebra and its applications*, 29:293–322, 1980.

Michael L Parks, Eric De Sturler, Greg Mackey, Duane D Johnson, and Spandan Maiti. Recycling krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5): 1651–1674, 2006.

John W Pearson and Jennifer Pestana. Preconditioners for krylov subspace methods: An overview. *GAMM-Mitteilungen*, 43(4):e202000015, 2020.

John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pp. 65–118. Elsevier, 1976.

Takao Sakurai, Takahiro Katagiri, Hisayasu Kuroda, Ken Naono, Mitsuyoshi Igai, and Satoshi Ohshima. A sparse matrix library with automatic selection of iterative solvers and preconditioners. *Procedia Computer Science*, 18:1332–1341, 2013.

Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. A survey of methods for automated algorithm configuration. *Journal of Artificial Intelligence Research*, 75:425–487, 2022.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Kirk M Soodhalter, Eric de Sturler, and Misha E Kilmer. A survey of subspace recycling iterative methods. *GAMM-Mitteilungen*, 43(4):e202000016, 2020.

Ziyuan Tang, Hong Zhang, and Jie Chen. Graph neural networks for selection of preconditioners and krylov solvers. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, pp. 1–11, 2022.

Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM international conference on management of data*, pp. 1009–1024, 2017.

Hong Wang, Zhongkai Hao, Jie Wang, Zijie Geng, Zhen Wang, Bin Li, and Feng Wu. Accelerating data generation for neural operators via Krylov subspace recycling. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=UpgRVWexaD.

Hantao Xiong, Wangdong Yang, Weiqing He, Shengle Lin, Keqin Li, and Kenli Li. Mm-autosolver: A multimodal machine learning method for the auto-selection of iterative solvers and preconditioners. *Journal of Parallel and Distributed Computing*, 205:105144, 2025.

Hantao Xiong, Wangdong Yang, Shengle Lin, Weiqing He, Keqin Li, and Kenli Li. Solverset: A large-scale benchmark dataset for the auto-selection of the optimal combination of iterative solvers and preconditioners. *Journal of Circuits, Systems and Computers*, 35(09):2650020, 2026.

Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2309.03409. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3339f19c5fcee3ad74502947a32be9e6-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Jae-Seung Yeom, Jayaraman J Thiagarajan, Abhinav Bhatele, Greg Bronevetsky, and Tzanio Kolev. Data-driven performance modeling of linear solvers for sparse matrices. In *2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pp. 32–42. IEEE, 2016.

Yury Zabegaev, Eirik Keilegavlen, Einar Iversen, and Inga Berre. Automated linear solver selection for simulation of multiphysics processes in porous media. *Computer Methods in Applied Mechanics and Engineering*, 426:117031, 2024.

Yury Zabegaev, Inga Berre, and Eirik Keilegavlen. Data-driven linear solver selection and performance tuning for multiphysics simulations in porous media. *Computational Science and Engineering*, 3(1):4, 2026.

Hong Zhang, Barry Smith, Satish Balay, Le Chen, Murat Keceli, Lois Curfman McInnes, and Junchao Zhang. An agentic evaluation framework for ai-generated scientific code in petsc. *arXiv preprint arXiv:2603.15976*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.