AGENTIC PROGRAM EVOLUTION OF EXECUTABLE NUMERICAL OPERATORS FOR PDE DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-dependent partial differential equation (PDE) simulations support forecasting and design in fluid dynamics, pattern formation, wave propagation, and many other physical systems. Although the governing equations may be known, computational budgets require them to be realized on finite grids and time steps. Accuracy and stability therefore depend on expert choices of derivative approximation, nonlinear discretization, filtering, operator splitting, time integration, and numerical constants, especially when relevant scales are unresolved. We introduce APEX (Agentic Program Evolution of eXecutable numerical operators), a framework that treats these coupled choices as a typed executable program and improves them through evaluator-guided evolution. A deterministic compiler rejects candidates that violate type or resource constraints, while a frozen evaluator supplies structured diagnostics of accuracy, stability, spectral behavior, and cost. A Program Evolution Graph retains successful and failed trials, and recurring subprograms are promoted to a reusable operator library. Our analysis characterizes finite-horizon error propagation, structured search state, and semantics-preserving operator abstraction. Experiments across canonical PDE families show that APEX improves long-horizon accuracy over learned neural operators while remaining competitive with task-specific coarse solvers. Combined library-and-history reuse improves a related PDE under the same target-side call budget. In one seven-call synthesis run on under-resolved 2D Navier–Stokes, APEX synthesizes a near-cutoff shadow-state program whose final error is $19.3\times$ lower than the standard coarse solver. APEX provides an explicit, auditable, and reusable approach to numerical-method design.

1 INTRODUCTION

Many scientific workflows begin with a known time-dependent partial differential equation (PDE), yet the equation alone does not determine a finite-resolution simulation. Practitioners must still choose derivative approximations, nonlinear forms, filters, operator splittings, time integrators, and numerical constants. Continuously equivalent formulations can therefore produce different truncation errors, resolved spectra, and long-horizon stability after discretization. Neural operators amortize repeated PDE solves by learning maps between function spaces (Lu et al., 2021; Li et al., 2021; Kovachki et al., 2023), while data-driven discretizations and message-passing solvers insert learned approximations into numerical time stepping (Bar-Sinai et al., 2019; Kochkov et al., 2021; Brandstetter et al., 2022). Together, these developments show that the numerical realization of known dynamics is itself a meaningful learning target.

Recent systems broaden this target with language-model-driven automation. CodePDE treats PDE solving as solver-code generation, AutoNumerics uses multiple agents to design and verify numerical solvers, and LLM-PDEveloper develops modules for existing PDE libraries (Li et al., 2026; Du et al., 2026; Wu et al., 2025). Other work composes classical and learned components into Pareto-optimal meta-solvers or transfers knowledge while configuring neural PDE surrogates (Lee et al., 2026; Wuwu et al., 2025). In parallel, evaluator-guided systems have established program evolution as a practical route to algorithmic discovery (Romera-Paredes et al., 2024; Novikov et al., 2025), and program-synthesis methods have learned reusable abstractions across related tasks (Ellis et al., 2021; Grayeli et al., 2024). These approaches optimize different computational objects: solver source code, configurations within a prescribed family, or general-purpose programs. This

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.

leaves a focused design question for known PDEs: how can an agent search over numerical structure while keeping every candidate executable, comparable under a fixed scientific protocol, and reusable across equations?

We introduce APEX (Agentic Program Evolution of eXecutable numerical operators), an agentic framework for synthesizing executable numerical operators. APEX represents a solver as a typed program $P = (G_s, G_t, \boldsymbol{\theta})$, comprising a spatial operator graph, a temporal program, and numerical constants. A deterministic compiler maps legal programs to executable kernels, and rejects invalid syntax, types, or resource use before evaluation. The agent cannot modify the PDE specification, reference data, compiler semantics, or evaluation protocol. It receives structured diagnostics that expose one-step and rollout error, gradient and spectral mismatch, stability, and cost, then proposes semantic edits to the program. A Program Evolution Graph records each hypothesis, edit, outcome, and interpretation, including failed experiments. Recurrent subgraphs can be abstracted into typed, finitely expandable operators in an evolving library and transferred to subsequent PDE tasks.

Contributions. This paper offers three contributions as follows:

- We formulate the numerical realization of a known PDE as constrained synthesis of an executable evolution operator, jointly exposing spatial discretization, temporal integration, and numerical constants to program-level search.
- We develop an agentic search framework that couples deterministic typed compilation with a frozen structured evaluator, evolution memory, and an operator library whose abstractions preserve executable semantics.
- We evaluate APEX on five PDE families, parameter extrapolation, fixed-call-budget mechanism ablations, and cross-PDE reuse. Most notably, one APEX synthesis run in a severely under-resolved Navier–Stokes (NS) regime produces a dynamic near-cutoff shadow state with cross-scale feedback: an interpretable augmented-state mechanism composed from generic primitives, not specified by the expert baseline or seeded as a composite operator.

2 Related Work

Learned representations for PDE simulation. Neural PDE models differ in the computational object they learn. DeepONet and FNO approximate solution operators, while multiscale graph operators and MeshGraphNets model interactions or dynamics on irregular discretizations (Lu et al., 2021; Li et al., 2021; 2020; Pfaff et al., 2021). Transolver introduces geometry-aware attention for general meshes (Wu et al., 2024). Other approaches learn solver components or corrections, including multigrid prolongation, differentiable solver-in-the-loop correction, and coarse turbulence dynamics (Greenfeld et al., 2019; Um et al., 2020; Stachenfeld et al., 2022). PDEBench standardizes evaluation across equations, while DPOT studies large-scale operator pretraining (Takamoto et al., 2022; Hao et al., 2024). Together, these works cover operator approximation, autoregressive simulation, learned corrections within classical solvers, and broad pretraining, but the admissible computation is fixed by the selected architecture or interface. APEX instead searches a typed executable program spanning spatial expressions, time integration, auxiliary state, and numerical constants; each discovered mechanism remains explicit and compiler-checked.

Automated solver and program discovery. CodePDE generates PDE solver code, while AutoPDE maintains an explicit, revisable numerical strategy during code-agent execution (Li et al., 2026; Dong et al., 2026). ADSL-PDE compiles a DSL of neural-solver design choices (Kong et al., 2026); APEX instead evolves explicit spatial–temporal operators and persistent state, with cross-PDE library/history reuse. The closest systems either generate a complete solver or optimize programs against a task-specific evaluator. Beyond PDEs, executable evaluation and evolutionary code search have supported discoveries in combinatorics, neural architecture design, matrix multiplication, sorting, and reward-code design (Romera-Paredes et al., 2024; Chen et al., 2023; Fawzi et al., 2022; Mankowitz et al., 2023; Ma et al., 2024). Stochastic superoptimization likewise searches executable instruction sequences (Schkufza et al., 2013). DreamCoder learns reusable abstractions across synthesis tasks, while LILO combines synthesis, compression, and documentation to construct interpretable libraries (Ellis et al., 2021; Grand et al., 2024). These results show that executable feedback can uncover algorithmic structure absent from expert seeds, while library induction

converts recurring solutions into reusable concepts. APEX combines both capabilities within a numerical DSL: a frozen evaluator returns structured scientific diagnostics, the evolution graph retains failed trials, and library entries remain finitely expandable into typed primitives.

3 METHOD

Figure 1 summarizes APEX as a closed loop from a known PDE to typed program synthesis, deterministic compilation, frozen numerical evaluation, and memory/library updates. Each evolution round uses the returned diagnostics to propose the next bounded batch while leaving the scientific protocol unchanged. The complete procedure is given in Appendix A.1.

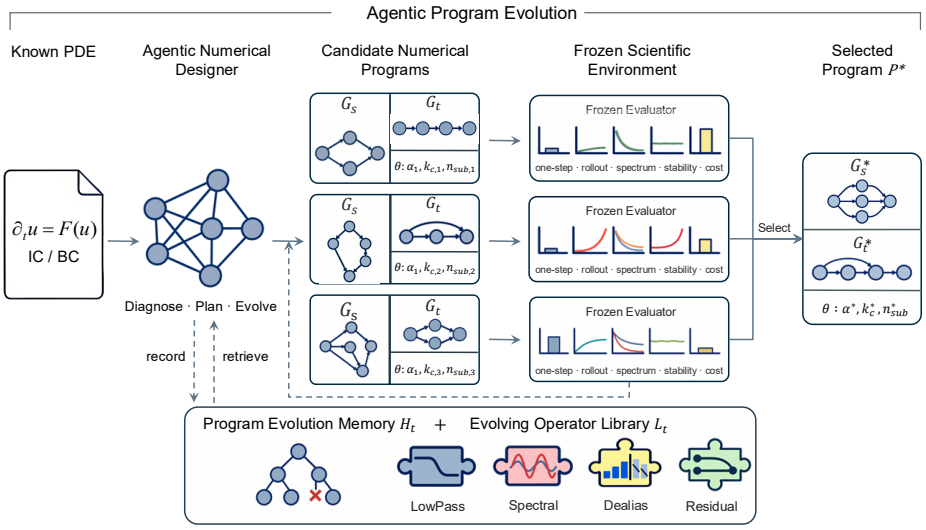


Figure 1: APEX evolves typed programs $(G_s, G_t, \boldsymbol{\theta})$ for a known PDE using frozen evaluation, evolution memory $\mathcal{H}_r$, and operator library $\mathcal{L}_r$, ultimately selecting $P^\star$.

3.1 PROBLEM FORMULATION AND EXECUTABLE OPERATOR PROGRAMS

We consider a known family of time-dependent PDEs

$$\partial_t \mathbf{u} = \mathcal{F}_{\mathcal{E}}(\mathbf{x}, t, \mathbf{u}, \nabla \mathbf{u}, \ldots; \boldsymbol{\eta}), \qquad \mathcal{B}[\mathbf{u}] = \mathbf{g}, \tag{1}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, $\boldsymbol{\eta} \in \Theta$, and $\mathcal{E} = (\mathcal{F}, \Omega, \mathcal{B}, \Theta)$ specifies the equation family, domain, boundary conditions, and admissible physical parameters. It induces an exact evolution operator $\mathcal{S}_{\Delta t}^{\mathcal{E},\boldsymbol{\eta}} : \mathbf{u}(t) \mapsto \mathbf{u}(t + \Delta t)$. APEX searches for a typed numerical program

$$P = (G_s, G_t, \boldsymbol{\theta}), \qquad \widehat{\mathcal{S}}_{P,\Delta t} = G_t[G_s; \boldsymbol{\theta}], \tag{2}$$

where the spatial graph G_s implements a discrete right-hand side $\widehat{\mathcal{F}}_P$, G_t advances it in time, and $\boldsymbol{\theta}$ contains numerical constants. The objective is to approximate $\mathcal{S}_{\Delta t}^{\mathcal{E},\boldsymbol{\eta}}$ over the prescribed parameter and initial-condition distributions at a fixed computational resolution. The equation is given throughout; APEX synthesizes its numerical realization and does not perform PDE discovery.

3.2 TYPED DSL AND DETERMINISTIC COMPILATION

Let $\mathcal{T}$ be a finite set of numerical types and $\mathcal{B}$ an immutable set of typed primitive operators. At evolution round r, the active operator language is $\mathcal{L}_r = \mathcal{B} \cup \mathcal{O}_r$, where $\mathcal{O}_r$ contains the reusable composite operators available before that round. A scratch run has $\mathcal{O}_0 = \emptyset$; a reuse run initializes $\mathcal{O}_0$ from the frozen source-library snapshot. Every $a \in \mathcal{L}_r$ has a signature $a : \tau_1 \times \cdots \times \tau_k \to \tau$ with $\tau_i, \tau \in \mathcal{T}$. Executable expressions are generated by the typed production rule

$$e^\tau = a(e^{\tau_1}, \ldots, e^{\tau_k}), \tag{3}$$

where $a : \tau_1 \times \cdots \times \tau_k \to \tau$. The spatial graph G_s and temporal program G_t in Eq. (2) are terms in this grammar. Their composition is therefore closed under the declared types.

For a fixed sandbox capacity $\mathcal{C}_{\text{sandbox}}$, we denote the admissible program space by $\mathfrak{P}_r$. A program $P = (G_s, G_t, \boldsymbol{\theta})$ belongs to $\mathfrak{P}_r$ only if it is well typed under $\mathcal{L}_r$, has a finite spatial DAG and bounded temporal loops, expands every composite operator finitely into primitives in $\mathcal{B}$, and satisfies the resource constraint $\text{Cost}(P) \preceq \mathcal{C}_{\text{sandbox}}$. Deterministic compilation is the total map

$$\text{Compile}_r(P) = \begin{cases} \widehat{\mathcal{S}}_{P,\Delta t}, & P \in \mathfrak{P}_r, \\ \perp_{\text{comp}}, & \text{otherwise}, \end{cases} \tag{4}$$

where $\perp_{\text{comp}}$ is a recorded compilation failure. This definition gives all later evaluation, memory, and abstraction operations a common executable interface. Appendix A.2 specifies the concrete types, primitive vocabulary, AST representation, legality checks, and compilation path.

3.3 Frozen Evaluator and Structured Diagnostics

Only P_r, the search memory $\mathcal{H}_r$, and the library $\mathcal{L}_r$ evolve. Before synthesis, we freeze the PDE specification, development trajectories $\mathcal{D}_{\text{dev}}$, data split, reference solver, rollout horizon, metrics, tolerances, compiler semantics, execution scripts, and resource counters. Every program is evaluated through the same interface

$$\mathbf{d}(P) = \text{Eval}_{\text{fixed}}(P; \mathcal{E}, \mathcal{D}_{\text{dev}}). \tag{5}$$

For trajectory m and rollout step h, let

$$e_{m,h}(P) = \frac{\|\widehat{\mathbf{u}}^P_{m,h} - \mathbf{u}^{\text{ref}}_{m,h}\|_2}{\|\mathbf{u}^{\text{ref}}_{m,h}\|_2 + \epsilon}, \quad E_H(P) = \frac{1}{N_{\text{traj}} H} \sum_{m=1}^{N_{\text{traj}}} \sum_{h=1}^{H} e_{m,h}(P), \tag{6}$$

with E_1 denoting one-step error. The gradient diagnostic emphasizes fronts and under-resolved small scales:

$$E_{\nabla,H}(P) = \frac{\sum_{m=1}^{N_{\text{traj}}} \sum_{h=1}^{H} \|\nabla \widehat{\mathbf{u}}^P_{m,h} - \nabla \mathbf{u}^{\text{ref}}_{m,h}\|_2}{\sum_{m=1}^{N_{\text{traj}}} \sum_{h=1}^{H} \|\nabla \mathbf{u}^{\text{ref}}_{m,h}\|_2 + \epsilon}. \tag{7}$$

For PDEs admitting a Fourier representation, the evaluator also compares spectral amplitudes:

$$E_{\text{spec},H}(P) = \frac{1}{N_{\text{traj}} H} \sum_{m=1}^{N_{\text{traj}}} \sum_{h=1}^{H} \frac{\big\| |\,\text{FFT}(\widehat{\mathbf{u}}^P_{m,h})| - |\,\text{FFT}(\mathbf{u}^{\text{ref}}_{m,h})| \big\|_2}{\big\| |\,\text{FFT}(\mathbf{u}^{\text{ref}}_{m,h})| \big\|_2 + \epsilon}. \tag{8}$$

This diagnostic is marked unavailable when the task has no appropriate Fourier representation. The evaluator further reports the raw state-growth ratio

$$G_H(P) = \frac{\max_{m,h} \|\widehat{\mathbf{u}}^P_{m,h}\|_2}{\max_{m,h} \|\mathbf{u}^{\text{ref}}_{m,h}\|_2 + \epsilon}, \tag{9}$$

a compilation or numerical-failure indicator I_{fail}, and $\mathbf{c}(P) = [T_{\text{wall}}, T_{\text{GPU}}, |V_P|]$. Thus $\mathbf{d}(P) = [E_1, E_H, E_{\nabla,H}, E_{\text{spec},H}, G_H, I_{\text{fail}}, \mathbf{c}(P)]$ describes both accuracy and computational behavior. The agent may read these diagnostics but cannot alter how they are produced.

3.4 Agentic Program Evolution

A deterministic initializer constructs

$$P_0 = \text{Init}_{\text{fixed}}(\mathcal{E}, \mathcal{L}_0), \tag{10}$$

which is evaluated and recorded as the root without an LLM call. We use r for the evolution round and reserve t for physical time. At round r, the agent observes the search state $S_r = (\mathcal{E}, P_r, \mathbf{d}_r, \mathcal{H}_r, \mathcal{L}_r, r, N_{\text{LLM}})$, comprising the PDE specification, current program and diagnostics, evolution memory, active operator language, round index, and total call budget. One LLM call applies a context-conditioned proposal policy,

$$\mathcal{Q}_r = \{P_r^{(k)}\}_{k=1}^{K_r} = \pi_{\text{agent}}(S_r), \tag{11}$$

to produce a bounded batch of sibling hypotheses. A proposal may edit an AST subgraph, add or remove a branch, change the time integrator, revise numerical constants, or replace the current construction. Structural and scalar choices are handled in this single process.

Each candidate is canonicalized, checked, compiled, and independently evaluated. The results

$$\mathcal{R}_r = \left\{ (\overline{P}_r^{(k)}, \mathbf{d}_r^{(k)}) \right\}_{k=1}^{K_r} \tag{12}$$

update the persistent state and determine P_{r+1}. Results must return before the next call, preventing an unbounded evaluation queue inside one round. Consequently, N_{LLM} calls yield the selected states $P_0, \ldots, P_{N_{\mathrm{LLM}}}$, while the number of executed and unique candidates is reported separately. Comparisons match adaptive proposal rounds and sandbox configuration, not executed-candidate or evaluator counts or total compute.

3.5 Evolution Memory and Reusable Operators

The Program Evolution Graph $\mathcal{H}_r = (\mathcal{V}_r, \mathcal{A}_r^H)$ stores complete program states as nodes, $v_i = (P_i, h(P_i), \mathbf{d}_i)$, with $h(P_i)$ the canonical AST hash. An edge records the hypothesis, semantic edit, parent and child diagnostics, outcome, and agent interpretation. Failed filters, unstable integrators, and ineffective parameter regimes therefore remain available as negative computational knowledge instead of disappearing from the search trace.

The learned portion of the operator language is $\mathcal{O}_r = \{O_i\}_{i=1}^{M_r}$, so the active language remains $\mathcal{L}_r = \mathcal{B} \cup \mathcal{O}_r$ as defined in Section 3.2. Each reusable operator stores

$$O_i = (A_i, \sigma_i, m_i), \tag{13}$$

where A_i is a canonical AST, σ_i its type signature, and m_i semantic metadata describing purpose, tested regimes, effects, and failure modes. The agent may CREATE, MERGE, or PRUNE components without hand-set trigger thresholds. Semantic validity is enforced by requiring every abstraction to expand finitely to the primitive DSL. Across source PDEs, each task receives an independent deterministic root while accumulated memory and operators follow the predeclared reuse protocol. For held-out evaluation, the source snapshot is frozen; task-local updates never propagate to another held-out PDE.

3.6 Search Protocol and Implementation Constraints

Canonicalization maps equivalent ASTs to $\overline{P}$ and a deterministic hash, allowing exact duplicates to reuse cached diagnostics. Evaluation runs in a network-isolated sandbox with read-only reference data and evaluator, fixed GPU/CPU/memory limits, and a single bounded batch per call. These controls constrain execution without prescribing the agent's edit policy.

Let $\mathcal{A}_{N_{\mathrm{LLM}}}$ contain the root and every valid candidate completed within budget. With J_{task} fixed before search and no early stopping, $P^\star \in \arg\min_{P \in \mathcal{A}_{N_{\mathrm{LLM}}},\, I_{\mathrm{fail}}(P)=0} J_{\mathrm{task}}(P)$. The returned executable operator is $\widehat{\mathcal{S}}^\star_{\Delta t} = \mathrm{Compile}(P^\star)$. Appendix A.1 gives the complete synthesis procedure.

4 Theoretical Analysis

We isolate three conditional properties that motivate APEX; complete statements and proofs appear in Appendix A.3.

Long-horizon evaluation. Let $u_{n+1} = \mathcal{S}(u_n)$ and $\widehat{u}_{n+1} = \widehat{\mathcal{S}}(\widehat{u}_n)$. Suppose both trajectories remain in a finite-rollout region $\mathcal{X}_N$, the uniform one-step discrepancy on this region is at most ε, and $\widehat{\mathcal{S}}$ is L-Lipschitz there. For $e_n = \|\widehat{u}_n - u_n\|$,

$$e_N \le L^N e_0 + \varepsilon \sum_{k=0}^{N-1} L^k. \tag{14}$$

A small one-step error can therefore coexist with rapid rollout-error growth when perturbations are amplified. This supports evaluating E_H and stability in addition to E_1; it is not a global stability guarantee.

Structured search state. Let Z_r^{c} contain only the current program and a scalar score, while Z_r^{a} additionally contains the PDE, structured diagnostics, evolution graph, library, and remaining call budget. If $Z_r^{\mathrm{c}} = g(Z_r^{\mathrm{a}})$ and fixed-rule policies form a subset of the admissible context-conditioned policies, then

$$R^\star(Z_r^{\mathrm{a}}) \le R^\star(Z_r^{\mathrm{c}}), \qquad \inf_{\pi\in\Pi_{\mathrm{agent}}} \mathbb{E}[J_{N_{\mathrm{LLM}}}(\pi)] \le \inf_{\pi\in\Pi_{\mathrm{fixed}}} \mathbb{E}[J_{N_{\mathrm{LLM}}}(\pi)]. \tag{15}$$

These are best-attainable decision and policy-class statements. They do not imply that a particular LLM uses the extra information optimally.

Safe operator abstraction. If a reusable operator O_Q deterministically expands to the same typed primitive AST Q, substituting it preserves executable semantics. Extending $\mathcal{L}$ with O_Q also cannot increase the shortest AST description:

$$\mathrm{Sem}(O_Q) = \mathrm{Sem}(Q), \qquad \ell_{\mathcal{L}\cup\{O_Q\}}(P) \le \ell_{\mathcal{L}}(P). \tag{16}$$

Repeated non-overlapping occurrences yield a corresponding node-count reduction. This establishes representational compression, not monotonic improvement in wall-clock search time.

5 Experiments

5.1 Experimental Setup

Benchmarks We construct a unified benchmark from five canonical PDE families: 1D Burgers, 1D Kuramoto–Sivashinsky (KS), 2D Navier–Stokes, 2D Gray–Scott (GS), and 3D acoustic wave dynamics. For each family, a high-resolution numerical solver generates trajectories under sampled initial conditions and one varying physical parameter; the trajectories are then restricted to a common observation grid. Each task contains 1,000 training, 200 validation, and 200 held-out trajectories with 41 snapshots, and every reference configuration passes an independent time-step refinement audit. Appendix A.4.1 specifies the equations, sampling procedure, grids, solvers, split construction, and audit tolerances.

Baselines We compare APEX with a task-specific standard coarse solver and four learned operators: FNO (Li et al., 2021), Transolver (Wu et al., 2024), LinearNO (Hu et al., 2026), and Tucker-FNO (Zhou et al., 2026). The coarse solvers combine established finite-difference, splitting, and time-integration schemes on the observation grid. Appendix A.4.2 gives their task-specific definitions and references. All baselines receive the same physical condition parameters and observation-grid states as APEX.

Evaluation Metrics We report one-step relative error R_1, 40-step rollout RMSE, final relative error R_{40}, numerical failures, and search cost. Learned-model uncertainty is the standard deviation over three training seeds, APEX uncertainty is the standard deviation over three independent synthesis runs, and deterministic coarse-solver uncertainty is estimated by bootstrap resampling held-out trajectories. Appendix A.4.3 gives the metric definitions and aggregation rules.

Implementation Learned operators use 100 epochs of five-step push-forward supervision and validation-only selection. APEX likewise selects one program on validation trajectories before a single held-out evaluation, so test data never enter search or model selection. Each logged LLM call is one API request with a bounded candidate batch; evaluating the deterministic root P_0 is zero-call. Appendices A.4.4 and A.4.5 detail the implementation, accounting, and identification limits.

5.2 Main Results

Table 1 shows that APEX obtains the lowest R_{40} on four of five PDEs. On 1D Burgers, APEX reaches $R_{40} = 6.44\times10^{-5}$, compared with 8.66×10^{-5} for the standard coarse solver and 1.59×10^{-2} for Tucker-FNO, the strongest learned baseline. On 2D NS, the corresponding values are 2.66×10^{-3} for APEX, 3.45×10^{-3} for the standard solver, and 1.45×10^{-2} for LinearNO. Results on 2D GS are more mixed: the standard solver has lower R_1 (1.66×10^{-4} versus 1.84×10^{-4}), whereas APEX lowers R_{40} from 1.25×10^{-3} to 1.01×10^{-3}. On 3D Wave, APEX records $R_{40} = 1.52\times10^{-3}$, close to the standard solver at 1.42×10^{-3} and below Tucker-FNO at 7.95×10^{-3}. Appendix A.7 shows that ten-run validation selection closes this default-budget gap.

Table 1: Mean errors on the five-PDE benchmark (lower is better). Best results are bold with gray shading, and second-best results are underlined. All values use normalized scientific notation with three significant digits. Appendix A.8 reports the corresponding standard deviations.

PDE	Metric	Standard coarse	FNO	Transolver	LinearNO	Tucker-FNO	APEX
1D Burgers	R_1	2.41×10^{-5}	1.28×10^{-2}	8.79×10^{-3}	4.97×10^{-3}	4.26×10^{-3}	$\mathbf{1.89\times10^{-5}}$
	RMSE	4.86×10^{-5}	2.21×10^{-2}	1.53×10^{-2}	8.81×10^{-3}	7.74×10^{-3}	$\mathbf{3.61\times10^{-5}}$
	R_{40}	8.66×10^{-5}	4.38×10^{-2}	3.06×10^{-2}	1.81×10^{-2}	1.59×10^{-2}	$\mathbf{6.44\times10^{-5}}$
1D KS	R_1	4.98×10^{-5}	1.76×10^{-2}	1.21×10^{-2}	7.48×10^{-3}	6.61×10^{-3}	$\mathbf{3.73\times10^{-5}}$
	RMSE	9.87×10^{-5}	3.06×10^{-2}	2.18×10^{-2}	1.37×10^{-2}	1.22×10^{-2}	$\mathbf{7.58\times10^{-5}}$
	R_{40}	2.17×10^{-4}	6.12×10^{-2}	4.51×10^{-2}	2.92×10^{-2}	2.61×10^{-2}	$\mathbf{1.64\times10^{-4}}$
2D NS	R_1	3.87×10^{-4}	3.13×10^{-3}	2.42×10^{-3}	1.86×10^{-3}	1.97×10^{-3}	$\mathbf{2.95\times10^{-4}}$
	RMSE	1.34×10^{-3}	1.07×10^{-2}	8.26×10^{-3}	6.34×10^{-3}	6.74×10^{-3}	$\mathbf{9.71\times10^{-4}}$
	R_{40}	3.45×10^{-3}	2.48×10^{-2}	1.89×10^{-2}	1.45×10^{-2}	1.53×10^{-2}	$\mathbf{2.66\times10^{-3}}$
2D GS	R_1	$\mathbf{1.66\times10^{-4}}$	1.42×10^{-3}	1.08×10^{-3}	8.65×10^{-4}	9.12×10^{-4}	1.84×10^{-4}
	RMSE	5.49×10^{-4}	4.71×10^{-3}	3.61×10^{-3}	2.96×10^{-3}	3.01×10^{-3}	$\mathbf{4.82\times10^{-4}}$
	R_{40}	1.25×10^{-3}	1.09×10^{-2}	8.22×10^{-3}	6.59×10^{-3}	6.89×10^{-3}	$\mathbf{1.01\times10^{-3}}$
3D Wave	R_1	$\mathbf{1.98\times10^{-4}}$	1.91×10^{-3}	1.49×10^{-3}	1.23×10^{-3}	1.04×10^{-3}	2.27×10^{-4}
	RMSE	$\mathbf{6.38\times10^{-4}}$	6.38×10^{-3}	4.97×10^{-3}	4.14×10^{-3}	3.56×10^{-3}	6.88×10^{-4}
	R_{40}	$\mathbf{1.42\times10^{-3}}$	1.46×10^{-2}	1.12×10^{-2}	9.25×10^{-3}	7.95×10^{-3}	1.52×10^{-3}

5.3 2D NS Parameter Extrapolation

All methods are evaluated zero-shot outside the training viscosity interval: learned operators receive no fine-tuning or OOD renormalization, while APEX changes only ν in the otherwise frozen program. Every method has higher errors at low viscosity, making this split empirically harder.

Table 2: Mean zero-shot viscosity-extrapolation errors on regular 2D NS (lower is better). Low/high OOD use $\nu \in [2.5\times10^{-4}, 5\times10^{-4})$ and $\nu \in (2\times10^{-3}, 3\times10^{-3}]$. Best/second-best results are bold/underlined; Appendix Table 7 reports the corresponding standard deviations.

Method	Low-viscosity OOD			High-viscosity OOD		
	R_1	RMSE	R_{40}	R_1	RMSE	R_{40}
Standard coarse	6.20×10^{-4}	1.93×10^{-3}	5.10×10^{-3}	$\mathbf{3.10\times10^{-4}}$	$\mathbf{1.05\times10^{-3}}$	$\mathbf{2.85\times10^{-3}}$
FNO	3.95×10^{-3}	1.39×10^{-2}	3.70×10^{-2}	3.30×10^{-3}	1.15×10^{-2}	2.70×10^{-2}
Transolver	3.05×10^{-3}	1.06×10^{-2}	2.83×10^{-2}	2.55×10^{-3}	8.70×10^{-3}	2.05×10^{-2}
LinearNO	2.42×10^{-3}	8.30×10^{-3}	2.22×10^{-2}	2.01×10^{-3}	6.90×10^{-3}	1.58×10^{-2}
Tucker-FNO	2.55×10^{-3}	8.60×10^{-3}	2.30×10^{-2}	2.10×10^{-3}	7.10×10^{-3}	1.63×10^{-2}
APEX	$\mathbf{5.20\times10^{-4}}$	$\mathbf{1.61\times10^{-3}}$	$\mathbf{4.55\times10^{-3}}$	3.40×10^{-4}	1.10×10^{-3}	3.00×10^{-3}

As shown in Table 2, on low viscosity, APEX lowers R_1, RMSE, and R_{40} relative to the standard coarse solver by 16.1%, 16.6%, and 10.8%, and its R_{40} is $4.9\times$ below LinearNO, the strongest learned baseline. At high viscosity, the coarse solver is modestly better ($R_{40} = 2.85\times10^{-3}$ versus 3.00×10^{-3}), while APEX remains $5.3\times$ below LinearNO. The standard deviations are small relative to the means. These results support zero-shot extrapolation by the frozen executable program, but not uniform dominance over the expert solver across viscosities.

5.4 Cross-PDE Knowledge Reuse

We transfer a frozen snapshot accumulated on Burgers and KS to a new generalized Burgers–KS family. Scratch synthesis receives only the primitive library; full experience reuse receives the combined learned library and evolution history. Both settings use the same frozen evaluator, sandbox,

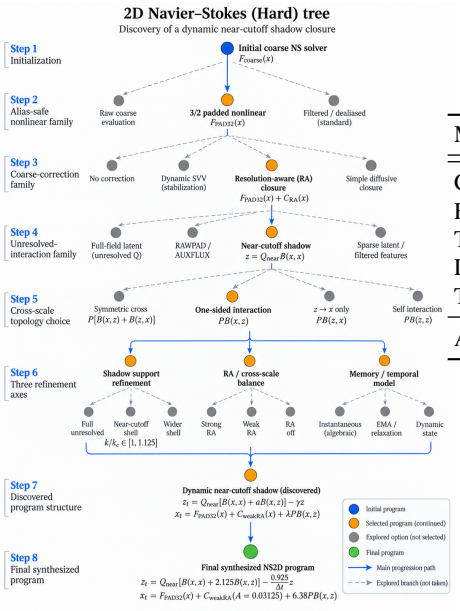
target objective, and seven target-side LLM calls. External physics and learned baselines contextualize the final accuracy.

Table 3: Generalized Burgers–KS results after seven target-side calls. Full reuse combines the frozen operator library and evolution history; lower is better. Best results are bold with gray shading, and second-best results are underlined.

Method	R_1	RMSE	R_{40}
WENO5+FD4+IMEX-CNAB2	$\underline{5.86\times10^{-5}}$	$\underline{1.28\times10^{-4}}$	$\underline{2.80\times10^{-4}}$
FNO	1.34×10^{-2}	2.39×10^{-2}	4.92×10^{-2}
Transolver	9.17×10^{-3}	1.65×10^{-2}	3.45×10^{-2}
LinearNO	5.86×10^{-3}	1.06×10^{-2}	2.24×10^{-2}
Tucker-FNO	5.17×10^{-3}	9.32×10^{-3}	1.96×10^{-2}
Scratch synthesis	1.21×10^{-4}	3.04×10^{-4}	7.20×10^{-4}
Full experience reuse	$\mathbf{4.39\times10^{-5}}$	$\mathbf{9.68\times10^{-5}}$	$\mathbf{2.05\times10^{-4}}$

From the rounded means in Table 3, full reuse reduces R_1, RMSE, and R_{40} relative to scratch by 63.7%, 68.2%, and 71.5%, and relative to the physics solver by 25.1%, 24.4%, and 26.8%. These comparisons identify combined experience reuse under a fixed target-side call budget; they separate neither library from memory nor candidate compute across calls (Appendix A.4.5). Appendix A.9 records the two source synthesis trees and the target tree initialized from their frozen snapshot.

5.5 Hard Under-Resolved 2D NS Discovery



Method	R_1	RMSE	R_{40}
Coarse spectral	5.66×10^{-2}	8.70×10^{-2}	1.42×10^{-1}
FNO	1.08×10^{-1}	1.83×10^{-1}	2.71×10^{-1}
Transolver	5.70×10^{-2}	8.29×10^{-2}	9.66×10^{-2}
LinearNO	3.99×10^{-2}	5.37×10^{-2}	6.17×10^{-2}
Tucker-FNO	$\underline{3.27\times10^{-2}}$	$\underline{4.10\times10^{-2}}$	$\underline{4.82\times10^{-2}}$
APEX shadow	$\mathbf{2.70\times10^{-3}}$	$\mathbf{4.89\times10^{-3}}$	$\mathbf{7.34\times10^{-3}}$

$$z_t = Q_{\text{near}}[B(\omega,\omega) + 2.125B(\omega, z)] - \frac{0.925}{\Delta t_{\text{save}}}z,$$
$$\omega_t = F_{\text{PAD32}}(\omega) + C_{\text{weakRA}}(\omega) + 6.38P_B(\omega, z).$$

Figure 2: Hard 2D NS discovery. Left: the seven-call semantic evolution from the deterministic root to the final program; dashed branches are evaluated alternatives. Right: held-out errors and the discovered augmented-state dynamics. The initial library contains a generic persistent auxiliary-state primitive but no near-cutoff shadow, NCROSS, or final composite.

The discovery setting uses intermittent-NS trajectories projected from 128^2 to 64^2. The selection objective is fixed before search as mean final-step error E_{40}, with the 95th percentile used only for tie-breaking. Fine-grid subgrid-enstrophy diagnostics are excluded from solver inputs, agent context, and candidate ranking. Starting from the deterministic zero-call root P_0, seven LLM calls produce the selected states $P_1, \ldots, P_7$ shown in Figure 2. Here B, Q_{near}, F_{PAD32}, C_{weakRA}, and

P_B denote bilinear NS interaction, near-cutoff projection, aliasing-controlled resolved dynamics, weak-RA correction, and shadow feedback, respectively. This is one synthesis trajectory; errors average 200 held-out physical trajectories and do not estimate rediscovery frequency.

The final program introduces a persistent near-cutoff shadow state z and feeds its cross-scale interaction back into the resolved vorticity. Its $R_{40} = 7.34\times10^{-3}$ is $19.3\times$ lower than the standard coarse solver and $6.5\times$ lower than Tucker-FNO, the strongest learned baseline. The evidence supports the complete program; without component-level ablations, it does not establish that the shadow state or any single feedback term independently causes the improvement.

5.6 Agent Mechanism Ablations

The ablations use regular 2D NS with the same model, prompt template, primitive library, LLM-call budget, data, and sandbox. Candidate executions are not forced to match because each policy chooses its own bounded batches. The three variants respectively make proposals open-loop, restrict edits to current-topology parameters, and hide the accumulated graph while retaining the current program and latest diagnostics.

Table 4: Agent ablations on regular 2D NS. Lower values are better. Best results are bold with gray shading, and second-best results are underlined.

Variant	R_1	RMSE	R_{40}
w/o evaluator feedback	1.18×10^{-2}	3.62×10^{-2}	8.74×10^{-2}
w/o structural evolution	4.38×10^{-3}	1.46×10^{-2}	3.71×10^{-2}
w/o evolution memory	$\underline{1.79\times10^{-3}}$	$\underline{5.92\times10^{-3}}$	$\underline{1.58\times10^{-2}}$
Full APEX	$\mathbf{2.98\times10^{-4}}$	$\mathbf{9.78\times10^{-4}}$	$\mathbf{2.60\times10^{-3}}$

Removing feedback, structural evolution, or memory raises R_{40} from 2.60×10^{-3} to 8.74×10^{-2}, 3.71×10^{-2}, and 1.58×10^{-2}; R_1 and RMSE degrade consistently. These results associate all three mechanisms with performance at the shared call budget. Unequal candidate counts preclude a compute-matched causal decomposition, and the always-active library is not isolated. Appendix A.6 separately compares APEX with random and evolutionary search in the same DSL under a matched executed-unique-candidate budget. Table 4 and Appendix Table 8 reuse a separate controlled run pool, explaining their small difference from Table 1.

6 Conclusion

We introduced APEX, which formulates the finite-resolution realization of known PDEs as constrained synthesis of typed executable numerical programs. Across canonical PDE families, the evolved programs improve long-horizon accuracy over learned neural operators and remain competitive with task-specific coarse solvers. Search experience transfers across related equations under a fixed call budget, and the hard under-resolved 2D NS experiment yields an interpretable augmented-state mechanism outside the seeded expert-solver family. Theoretical results clarify error propagation, the information available to structured search policies, and semantics-preserving abstraction, while ablations associate feedback, structural evolution, and memory with performance under the evaluated call-budget operating point. Current evidence is limited to the studied PDE families, structured grids, and fixed search budgets. It also does not isolate every learned-library component or term in the discovered NS program. Extending the language to irregular geometries and coupled multiphysics systems, and testing reusable abstractions across discretizations and resolutions, are important next steps.

AI USE STATEMENT

In the reported research, the OpenAI Responses API with `gpt-5.6-sol` was used within APEX to propose typed numerical-program hypotheses and edits; deterministic local code compiled, executed, and evaluated every candidate. OpenAI Codex was also used for literature search and citation checking, manuscript drafting and revision (including the presentation of mathematical claims, proofs, and experimental results), and LaTeX and table formatting. Generative AI was not used to generate reference trajectories, execute numerical experiments, or determine reported metric values. The authors reviewed the AI-assisted material against the cited sources, derivations, implementation, and experiment records, and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work uses synthetically generated PDE trajectories and involves no human participants, personal data, or sensitive attributes. Its primary foreseeable risks are computational resource use and inappropriate reliance on automatically synthesized numerical programs in safety-critical simulations. Typed compilation, fixed resource limits, network-isolated execution, and deterministic evaluation reduce execution and integrity risks. Discovered programs should nevertheless be independently validated at appropriate resolutions and against domain-specific physical constraints before deployment. We adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Section 3 defines the program representation, compiler, evaluator, and evolution protocol, and Section 5 states the common selection and evaluation procedure. Appendix A.1 gives pseudocode; Appendices A.2 and A.3 provide implementation details, assumptions, and complete proofs. Appendix A.4 specifies benchmark generation, solvers, data splits, metrics, model configurations, search budgets, and logging, while Appendix A.8 reports complete results and Appendix A.9 records the reuse trajectories. Together, these materials document the information needed to reconstruct the reported pipeline and comparisons.

REFERENCES

Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995. doi: 10.1137/0732037. URL https://doi.org/10.1137/0732037.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1814058116.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Angelica Chen, David M. Dohan, and David R. So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0342. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/184c1e18d00d7752805324da48ad25be-Abstract-Conference.html.

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. doi: 10.48550/arXiv.2606.10752. URL https://arxiv.org/abs/2606.10752.

Jianda Du, Youran Sun, and Haizhao Yang. AutoNumerics: An autonomous, PDE-agnostic multi-agent pipeline for scientific computing. *arXiv preprint arXiv:2602.17607*, 2026. doi: 10.48550/arXiv.2602.17607. URL https://arxiv.org/abs/2602.17607.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake–sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 835–850. ACM, 2021. doi: 10.1145/3453483.3454080. URL https://doi.org/10.1145/3453483.3454080.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://doi.org/10.1038/s41586-022-05172-4.

Gabriel Grand, Lionel Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4gYLottfsf.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1419.

Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2415–2423. PMLR, 2019. URL https://proceedings.mlr.press/v97/greenfeld19a.html.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-Regressive Denoising Operator Transformer for Large-Scale PDE Pre-Training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635. PMLR, 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Wenjie Hu, Sidun Liu, Peng Qiao, Zhenglun Sun, and Yong Dou. Transolver Is a Linear Transformer: Revisiting physics-attention through the lens of linear attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(1):408–416, 2026. doi: 10.1609/aaai.v40i1.37003. URL https://doi.org/10.1609/aaai.v40i1.37003.

Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. doi: 10.1006/jcph.1996.0130. URL https://doi.org/10.1006/jcph.1996.0130.

Aly-Khan Kassam and Lloyd N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005. doi: 10.1137/S1064827502410633. URL https://doi.org/10.1137/S1064827502410633.

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2101784118.

Shengxin Kong, Liwen Xu, and Jingwen Fu. Improving auto-design of neural PDE solvers with a domain-specific language. *arXiv preprint arXiv:2608.04384*, 2026. doi: 10.48550/arXiv.2608.04384. URL https://arxiv.org/abs/2608.04384.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

Youngkyu Lee, Shanqing Liu, Jérôme Darbon, and George Em Karniadakis. Automatic discovery of optimal meta-solvers for time-dependent nonlinear pdes. *Journal of Computational Physics*, 567:115313, 2026. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2026.115313. URL https://www.sciencedirect.com/science/article/pii/S0021999126006625.

Shanda Li, Tanya Marwah, Junhong Shen, Weiwei Sun, Andrej Risteski, Yiming Yang, and Ameet Talwalkar. Codepde: An inference framework for llm-driven pde solver generation. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=eG3Qy5Oux6.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4b21cf96d4cf612f239a6c322b10c8fe-Abstract.html.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618:257–263, 2023. doi: 10.1038/s41586-023-06004-9. URL https://doi.org/10.1038/s41586-023-06004-9.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. doi: 10.48550/arXiv.2506.13131. URL https://arxiv.org/abs/2506.13131.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Eric Schkufza, Rahul Sharma, and Alex Aiken. Stochastic superoptimization. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 305–316. ACM, 2013. doi: 10.1145/2451116.2451150. URL https://doi.org/10.1145/2451116.2451150.

Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988. doi: 10.1016/0021-9991(88)90177-5. URL https://doi.org/10.1016/0021-9991(88)90177-5.

Kimberly Stachenfeld, Drummond B. Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. In *International Conference on Learning Representations*, 2022.

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968. doi: 10.1137/0705041. URL https://doi.org/10.1137/0705041.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBENCH: An Extensive Benchmark for Scientific Machine Learning. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks*, 2022.

Kiwon Um, Robert Brand, Yun Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html.

Jean Virieux. P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 51(4):889–901, 1986. doi: 10.1190/1.1442147. URL https://doi.org/10.1190/1.1442147.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024.

Haoyang Wu, Xinxin Zhang, and Lailai Zhu. Automated code development for PDE solvers using large language models. *arXiv preprint arXiv:2509.25194*, 2025. doi: 10.48550/arXiv.2509.25194. URL https://arxiv.org/abs/2509.25194.

Qingpo Wuwu, Chonghan Gao, Tianyu Chen, Yihang Huang, Yuekai Zhang, Jianing Wang, Jianxin Li, Haoyi Zhou, and Shanghang Zhang. PINNsAgent: Automated PDE surrogation with large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68143–68165. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wuwu25a.html.

Guancheng Zhou, Zelin Zeng, Yisi Luo, Qi Xie, and Deyu Meng. Tucker-FNO: Tensor tucker-fourier neural operator and its universal approximation theory. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=UJvkXnuozY.

A APPENDIX

A.1 APEX SYNTHESIS PROCEDURE

Algorithm 1 APEX Synthesis

Require: PDE specification $\mathcal{E}$; initial operator library $\mathcal{L}_0$; fixed compiler and evaluator $\mathrm{Eval}_{\mathrm{fixed}}$; task objective J_{task}; LLM-call budget N_{LLM}

$P_0 \leftarrow \mathrm{Init}_{\mathrm{fixed}}(\mathcal{E}, \mathcal{L}_0)$
$d_0 \leftarrow \mathrm{Eval}_{\mathrm{fixed}}(P_0; \mathcal{L}_0)$
$\mathcal{H}_0 \leftarrow \mathrm{RecordRoot}(P_0, d_0)$
$\mathcal{R}_0 \leftarrow \{(P_0, d_0)\}$
$A_0 \leftarrow P_0$
$\mathcal{C} \leftarrow \{P_0\}$
for $r = 1, \ldots, N_{\mathrm{LLM}}$ **do**
  $\mathcal{S}_r \leftarrow (\mathcal{E}, A_{r-1}, \mathcal{R}_{r-1}, \mathcal{H}_{r-1}, \mathcal{L}_{r-1}, r, N_{\mathrm{LLM}})$
  $(A_r, \Delta\mathcal{L}_r, \mathcal{Q}_r) \leftarrow \pi_{\mathrm{agent}}(\mathcal{S}_r)$ {one LLM call}
  $\mathcal{L}_r \leftarrow \mathrm{UpdateLibrary}(\mathcal{L}_{r-1}, \Delta\mathcal{L}_r)$
  $\mathcal{R}_r \leftarrow \mathrm{CompileAndEval}_{\mathrm{fixed}}(\mathcal{Q}_r; \mathcal{L}_r)$
  $\mathcal{H}_r \leftarrow \mathrm{UpdateGraph}(\mathcal{H}_{r-1}, A_r, \Delta\mathcal{L}_r, \mathcal{Q}_r, \mathcal{R}_r)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathrm{ValidCompleted}(\mathcal{R}_r)$
end for
$P^\star \leftarrow \arg \min_{P \in \mathcal{C},\, I_{\mathrm{fail}}(P)=0} J_{\mathrm{task}}(P)$
return $P^\star$

A.2 TYPED DSL AND COMPILER IMPLEMENTATION

Programs are serialized as canonical JSON ASTs and rendered in the paper with a compact DSL. The concrete type set is

$$\mathcal{T} = \{\texttt{Field}[d, N_c], \texttt{SpectralField}[d, N_c], \texttt{Scalar}, \texttt{RHSOperator}, \texttt{TimeStepper}\},$$

where d is the spatial dimension and N_c is the number of state channels. The immutable primitive basis $\mathcal{B}$ is organized as follows.

Category	Primitive operators
Arithmetic	ADD, SUBTRACT, MULTIPLY, SCALE, POWER
Differential	DERIVATIVE, GRADIENT, DIVERGENCE, LAPLACIAN
Spectral	FFT, IFFT, MASK, SHIFT
Local/pointwise	PAD, CROP, CONVOLUTION, EXP, LOG, SIN, COS, TANH, ABS
Reduction/structure	MEAN, SUM, NORM, BRANCH, MERGE, GATE, RESIDUAL
Temporal/state	EULER, HEUN, SSPRK3, RK4, SUBSTEP, SPLIT, AUXILIARYSTATE

For example, $\mathrm{FFT} : \texttt{Field} \to \texttt{SpectralField}$, $\mathrm{Multiply} : \texttt{Field} \times \texttt{Field} \to \texttt{Field}$, and a time stepper consumes a field, a typed RHS operator, and Δt. Composition can create operators not present in $\mathcal{B}$, such as

$$\mathrm{LowPass} = \mathrm{IFFT} \circ \mathrm{Mask} \circ \mathrm{FFT}, \qquad \widehat{\mathcal{F}}_P(u) = \mathrm{LowPass}[-uD_x u] + \nu D_{xx} u. \tag{17}$$

The example illustrates that the PDE supplies the physical terms, while derivative realizations, nonlinear forms, filter placement, and time integration remain program choices.

Before lowering an AST, the compiler performs type and shape inference, verifies finite spatial DAGs and bounded temporal loops, rejects non-finite scalar values, recursively expands every $O_i \in \mathcal{O}_r$ into primitives in $\mathcal{B}$, and checks the estimated resource demand against $\mathcal{C}_{\mathrm{sandbox}}$. Network access, system calls, and dynamically generated raw Python are unavailable to the program. A legal AST is lowered in deterministic topological order to executable kernels; any failed check returns $\perp_{\mathrm{comp}}$ and is recorded as an evaluated search outcome.

A.3 Theoretical Assumptions and Complete Proofs

This section makes the conditions behind Section 4 explicit. The results explain the roles of rollout diagnostics, structured feedback, and the evolving operator library. They do not assert unconditional convergence of APEX or optimality of its language model.

A.3.1 Finite-Horizon Error Propagation

Let $(\mathcal{X}, \|\cdot\|)$ be a normed state space. Write $\mathcal{S} : \mathcal{X} \to \mathcal{X}$ for the exact one-step evolution operator and $\widehat{\mathcal{S}} : \mathcal{X} \to \mathcal{X}$ for a compiled numerical operator. The exact and numerical trajectories satisfy

$$u_{n+1} = \mathcal{S}(u_n), \qquad \widehat{u}_{n+1} = \widehat{\mathcal{S}}(\widehat{u}_n). \tag{18}$$

Theorem A.1 (Finite-horizon rollout bound)**.** *Fix $N \geq 1$, and let $\mathcal{X}_N \subseteq \mathcal{X}$ contain $\{u_n, \widehat{u}_n : 0 \leq n \leq N\}$. Assume that*

$$\sup_{u \in \mathcal{X}_N} \|\widehat{\mathcal{S}}(u) - \mathcal{S}(u)\| \leq \varepsilon, \tag{19}$$

$$\|\widehat{\mathcal{S}}(u) - \widehat{\mathcal{S}}(v)\| \leq L\|u - v\| \quad \textit{for all } u, v \in \mathcal{X}_N, \tag{20}$$

where $\varepsilon \geq 0$ and $L \geq 0$. Then, for $e_n = \|\widehat{u}_n - u_n\|$,

$$e_N \leq L^N e_0 + \varepsilon \sum_{k=0}^{N-1} L^k. \tag{21}$$

Proof. For each $0 \leq n < N$, both u_n and $\widehat{u}_n$ belong to $\mathcal{X}_N$. The triangle inequality, followed by equation 20 and equation 19, gives

$$\begin{aligned} e_{n+1} &= \|\widehat{\mathcal{S}}(\widehat{u}_n) - \mathcal{S}(u_n)\| \\ &\leq \|\widehat{\mathcal{S}}(\widehat{u}_n) - \widehat{\mathcal{S}}(u_n)\| + \|\widehat{\mathcal{S}}(u_n) - \mathcal{S}(u_n)\| \\ &\leq L e_n + \varepsilon. \end{aligned} \tag{22}$$

We prove the claimed expression by induction. For $N = 1$, equation 22 yields $e_1 \leq L e_0 + \varepsilon$. If $e_n \leq L^n e_0 + \varepsilon \sum_{k=0}^{n-1} L^k$, then

$$e_{n+1} \leq L^{n+1} e_0 + \varepsilon \sum_{k=1}^{n} L^k + \varepsilon = L^{n+1} e_0 + \varepsilon \sum_{k=0}^{n} L^k.$$

Thus equation 21 holds for every finite N. □

Corollary A.2 (Matched initial state)**.** *If $e_0 = 0$, then*

$$e_N \leq \begin{cases} \varepsilon(1 - L^N)/(1 - L), & 0 \leq L < 1, \\ N\varepsilon, & L = 1, \\ \varepsilon(L^N - 1)/(L - 1), & L > 1. \end{cases}$$

Proof. Set $e_0 = 0$ in Theorem A.1 and evaluate the finite geometric sum. The expression is valid at $L = 0$ and all three cases are nonnegative. □

A one-step counterexample. A measured one-step score at the initial state does not control later states. Let $\mathcal{X} = \mathbb{R}$, $\mathcal{S}(u) = 0$, $u_0 = \widehat{u}_0 = 0$, and choose $\varepsilon > 0$ and $L > 1$. Consider

$$\widehat{\mathcal{S}}_A(u) = \varepsilon, \qquad \widehat{\mathcal{S}}_B(u) = Lu + \varepsilon.$$

Both candidates have the same initial one-step error ε. Candidate A has $\widehat{u}_n^A = \varepsilon$ for every $n \geq 1$, whereas

$$\widehat{u}_n^B = \varepsilon \sum_{k=0}^{n-1} L^k = \varepsilon \frac{L^n - 1}{L - 1}.$$

The example does not impose the uniform consistency condition equation 19; it shows why an empirical E_1 evaluated only at initial states cannot replace rollout diagnostics. Theorem A.1 gives the complementary conditional guarantee once local consistency and perturbation amplification are controlled on the visited region.

A.3.2 Information and Policy-Class Inclusion

Let ξ_r be the latent search state at round r, $\mathcal{U}$ the set of admissible finite program-design actions, and $\ell(a, \xi_r)$ a pre-specified decision loss. Assume the loss is measurable, bounded below, and has a well-defined expectation for every policy considered. For an observation Z, let $\Delta(Z)$ denote the set of measurable decision rules that choose actions using only Z, and define

$$R^\star(Z) = \inf_{\delta \in \Delta(Z)} \mathbb{E}[\ell(\delta(Z), \xi_r)]. \tag{23}$$

Proposition A.3 (Value of a refined search observation)**.** *If $Z_r^{\mathrm{c}} = g(Z_r^{\mathrm{a}})$ almost surely for a measurable map g, then*

$$R^\star(Z_r^{\mathrm{a}}) \le R^\star(Z_r^{\mathrm{c}}).$$

Proof. Take any $\delta_{\mathrm{c}} \in \Delta(Z_r^{\mathrm{c}})$. The composition $\delta_{\mathrm{a}} = \delta_{\mathrm{c}} \circ g$ is measurable with respect to Z_r^{a} and satisfies $\delta_{\mathrm{a}}(Z_r^{\mathrm{a}}) = \delta_{\mathrm{c}}(Z_r^{\mathrm{c}})$ almost surely. Hence it has the same expected loss. Every coarse-observation rule therefore has an equivalent rule in $\Delta(Z_r^{\mathrm{a}})$, so $\Delta(Z_r^{\mathrm{c}}) \subseteq \Delta(Z_r^{\mathrm{a}})$ after this identification. Taking the infimum over the larger set proves the claim. □

The inequality can be strict. For example, suppose a coarse observation value merges two positive-probability fine states with distinct unique optimal actions, and choosing the wrong action incurs a positive loss gap. A rule based on Z_r^{a} can select the appropriate action in both states. A rule based only on the shared coarse value must use the same action and incurs the gap on at least one event. In APEX, equal scalar rollout scores can hide different spectral and state-growth diagnostics; the strictness condition applies only when those differences genuinely change the optimal next edit.

For the policy-class comparison, let $\mathcal{A}_{N_{\mathrm{LLM}}}(\pi)$ be the root and all valid candidates completed by policy π under the fixed LLM-call budget and sandbox. Define

$$J_{N_{\mathrm{LLM}}}(\pi) = \min_{\substack{P \in \mathcal{A}_{N_{\mathrm{LLM}}}(\pi) \\ I_{\mathrm{fail}}(P)=0}} J_{\mathrm{task}}(P), \tag{24}$$

with value $+\infty$ if the feasible set is empty.

Proposition A.4 (Policy-class inclusion)**.** *If every fixed-rule policy is representable by an admissible context-conditioned policy, so that $\Pi_{\mathrm{fixed}} \subseteq \Pi_{\mathrm{agent}}$, then*

$$\inf_{\pi \in \Pi_{\mathrm{agent}}} \mathbb{E}[J_{N_{\mathrm{LLM}}}(\pi)] \le \inf_{\pi \in \Pi_{\mathrm{fixed}}} \mathbb{E}[J_{N_{\mathrm{LLM}}}(\pi)].$$

Proof. The objective, call budget, sandbox, and treatment of failed candidates are identical for both policy classes. The set over which the left infimum is taken contains the set for the right infimum. An infimum over a superset cannot exceed the infimum over its subset. □

Propositions A.3 and A.4 concern attainable optima over idealized rule classes. They neither show that additional diagnostics are always relevant nor guarantee that the instantiated LLM finds a better policy. The ablations test whether the implementation realizes the predicted advantage under the experimental budget.

A.3.3 Semantics-Preserving Operator Abstraction

Let Q be a legal typed primitive AST subgraph with numerical semantics $\mathrm{Sem}(Q)$. A library operator O_Q is *safe* when the compiler replaces each call to O_Q, with identical parameters and wiring, by the same finite AST Q before execution. We also assume that this expansion and subsequent canonicalization preserve the primitive execution order and types.

Lemma A.5 (Semantic preservation)**.** *For every legal typed program context $C[\cdot]$,*

$$\mathrm{Sem}(C[O_Q]) = \mathrm{Sem}(C[Q]).$$

Consequently, replacing any collection of non-overlapping occurrences of Q by O_Q leaves the compiled evolution operator unchanged.

Proof. By the definition of a safe operator, macro expansion maps $C[O_Q]$ to the same primitive AST as $C[Q]$. The deterministic compiler therefore receives identical typed nodes, parameters, edges, and execution order. It produces the same primitive operations and numerical mapping. Applying the argument to non-overlapping occurrences one at a time proves the second statement. □

For a library $\mathcal{L}$, define the shortest semantic description of program P by

$$\ell_{\mathcal{L}}(P) = \min_{R:\mathrm{Sem}(R)=\mathrm{Sem}(P)} |R|, \tag{25}$$

where R ranges over legal representations using $\mathcal{L}$ and $|R|$ is its AST node count. A representation exists because the primitive expansion of P is legal, and the nonnegative integer node counts therefore admit a minimum.

Proposition A.6 (Description-length non-increase)**.** *Let $|Q| = m > 1$ and $\mathcal{L}' = \mathcal{L} \cup \{O_Q\}$, where one call to O_Q counts as one AST node. Then, for every legal P,*

$$\ell_{\mathcal{L}'}(P) \le \ell_{\mathcal{L}}(P).$$

If a shortest representation under $\mathcal{L}$ contains n_{occ} pairwise non-overlapping occurrences of Q, then

$$\ell_{\mathcal{L}'}(P) \le \ell_{\mathcal{L}}(P) - n_{\mathrm{occ}}(m-1).$$

Proof. Because $\mathcal{L} \subseteq \mathcal{L}'$, every representation legal under $\mathcal{L}$ remains legal under $\mathcal{L}'$. Minimizing over this superset proves the first inequality. For the second, choose a shortest old representation $R^\star$. Replace each of the n_{occ} non-overlapping copies of Q by one O_Q node. Lemma A.5 preserves the program semantics. Each replacement changes the node count by $1-m$, and non-overlap makes these reductions additive. The resulting legal representation has length $\ell_{\mathcal{L}}(P) - n_{\mathrm{occ}}(m-1)$; the new minimum cannot be larger. □

The proposition concerns representation length. Enlarging a library can also enlarge the branching factor seen by the agent, so it does not imply a monotone reduction in candidate executions or wall-clock time. Cross-PDE reuse experiments evaluate that separate empirical question.

A.4 Experimental Details

A.4.1 Benchmark Construction

All five main-benchmark datasets are generated specifically for this study; no public benchmark trajectories are reused. For each PDE family, we independently sample a physical-condition parameter and an initial condition, integrate the corresponding system on a high-resolution reference grid, and restrict the resulting states to the observation grid accessed by every evaluated method. Training, validation, and test splits contain 1,000, 200, and 200 trajectories, respectively, generated from disjoint random seeds. Each trajectory contains 41 equally spaced snapshots, defining 40 autoregressive prediction steps. Unless stated otherwise, all equations use periodic boundary conditions. The physical-condition parameter is sampled independently for every trajectory and provided explicitly to all methods. Table 5 summarizes the common construction parameters.

Table 5: Construction of the five-PDE benchmark. The grid arrow denotes restriction from the high-resolution reference grid to the observation grid used by all evaluated methods.

PDE	Condition parameter	Domain	Grid	T_{end}	Δt_{save}	Reference integrator
1D Burgers	$\nu \sim \mathcal{U}[10^{-3}, 4\times10^{-3}]$	$[0,1]$	$1024 \to 256$	2	0.05	pseudo-spectral ETDRK4, $\Delta t = 2.5\times10^{-4}$
1D KS	$\alpha \sim \mathcal{U}[0.8, 1.2]$	$[0,22]$	$1024 \to 256$	20	0.5	pseudo-spectral ETDRK4, $\Delta t = 0.025$
2D NS	$\nu \sim \mathcal{U}[5\times10^{-4}, 2\times10^{-3}]$	$[0,2\pi)^2$	$512^2 \to 256^2$	2	0.05	pseudo-spectral CNAB2, $\Delta t = 0.0025$
2D GS	$F \sim \mathcal{U}[0.025, 0.045]$	$[-1,1]^2$	$512^2 \to 256^2$	400	10	pseudo-spectral ETDRK4, $\Delta t = 1$
3D Wave	$\eta \sim \mathcal{U}[0.15, 0.35]$	$[0,1]^3$	$64^3 \to 32^3$	0.8	0.02	pseudo-spectral RK4, $\Delta t = 0.002$

Reference-to-observation restriction All trajectories are first evolved entirely on the stated high-resolution reference grid. For the periodic spectral problems, restriction is performed by centered Fourier truncation: Fourier coefficients outside the observation-grid Nyquist range are discarded and the retained coefficients are transformed back to the lower-resolution physical grid with the corresponding normalization. For multichannel systems, the same operation is applied independently to each state channel. Consequently, no evaluated method has access to high-resolution states or unresolved Fourier coefficients during training, search, or inference.

1D Burgers We solve the periodic viscous conservation law

$$u_t + \partial_x\left(\frac{u^2}{2}\right) = \nu u_{xx}, \qquad x \in [0,1], \tag{26}$$

where $\nu \sim \mathcal{U}[10^{-3}, 4{\times}10^{-3}]$. Initial conditions are sampled as smooth random periodic fields

$$u_0(x) = \sum_{k=1}^{8} \frac{a_k \cos(2\pi kx) + b_k \sin(2\pi kx)}{1+k^2}, \tag{27}$$

with $a_k, b_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. The spatial mean is removed and the field is rescaled to unit root-mean-square (RMS) amplitude before integration. Reference trajectories are generated on 1,024 grid points using a pseudo-spectral discretization with the two-thirds dealiasing rule and ETDRK4 (Kassam & Trefethen, 2005) with internal time step $\Delta t = 2.5{\times}10^{-4}$. The reference states are restricted to 256 points and saved every 0.05 time units until $T_{\text{end}} = 2$.

1D KS We consider the parameterized dynamics

$$u_t + uu_x + \alpha u_{xx} + u_{xxxx} = 0, \qquad x \in [0, 22], \tag{28}$$

where $\alpha \sim \mathcal{U}[0.8, 1.2]$. The pre-burn-in state is sampled as a zero-mean band-limited random Fourier field,

$$u_{\text{raw}}(x) = \sum_{k=1}^{8} \frac{a_k \cos(2\pi kx/22) + b_k \sin(2\pi kx/22)}{1+k^2}, \tag{29}$$

with independent standard-normal coefficients. It is normalized to unit RMS amplitude and then evolved for 20 time units before any benchmark state is recorded. This burn-in reduces sensitivity to the arbitrary raw initialization and places the saved trajectories near the parameter-dependent KS attractor. References use 1,024 Fourier modes, two-thirds dealiasing, and ETDRK4 with $\Delta t = 0.025$ before restriction to 256 observation points. Forty prediction intervals are saved with $\Delta t_{\text{save}} = 0.5$.

2D NS We use the periodic vorticity formulation

$$\omega_t + \mathbf{u}\cdot\nabla\omega = \nu\Delta\omega + f, \tag{30}$$

$$-\Delta\psi = \omega, \tag{31}$$

$$\mathbf{u} = (\partial_y\psi, -\partial_x\psi), \tag{32}$$

on $[0, 2\pi)^2$, with

$$f(x,y) = 0.1[\sin(4y) + \cos(4x)] \tag{33}$$

and $\nu \sim \mathcal{U}[5{\times}10^{-4}, 2{\times}10^{-3}]$. Initial vorticity fields are sampled spectrally. For each nonzero wavevector $\mathbf{k} = (k_x, k_y)$ satisfying $1 \le \|\mathbf{k}\|_2 \le 8$, we draw independent complex Gaussian coefficients and apply the smooth envelope

$$\widehat{\omega}_0(\mathbf{k}) \propto \frac{\xi_{\mathbf{k}}}{(1+\|\mathbf{k}\|_2^2)^{3/2}}, \tag{34}$$

with Hermitian symmetry imposed so that ω_0 is real valued. The zero mode is set to zero and each field is normalized to unit RMS vorticity.

Reference trajectories use a 512^2 pseudo-spectral discretization with two-thirds dealiasing. The viscous term is treated implicitly and the nonlinear and forcing terms explicitly using second-order Crank–Nicolson/Adams–Bashforth (CNAB2) time integration with internal step size $\Delta t = 0.0025$. States are restricted to the 256^2 observation grid and saved every 0.05 time units until $T_{\text{end}} = 2$. The sampled viscosity is supplied to every method, whereas no fine-grid state is available during training, synthesis, or evaluation.

2D GS We simulate

$$u_t = D_u \Delta u - uv^2 + F(1-u), \tag{35}$$

$$v_t = D_v \Delta v + uv^2 - (F+k)v, \tag{36}$$

on the periodic domain $[-1,1]^2$, where

$$D_u = 2{\times}10^{-5}, \qquad D_v = 10^{-5}, \qquad k = 0.060, \tag{37}$$

and $F \sim \mathcal{U}[0.025, 0.045]$. Each trajectory starts from the homogeneous state $(u,v) = (1,0)$ and receives $n_s \sim \mathcal{U}\{2,3,4,5\}$ localized perturbation seeds. For seed j, the center $\mathbf{c}_j$ is sampled uniformly from $[-0.75, 0.75]^2$, the width is sampled as $\sigma_j \sim \mathcal{U}[0.04, 0.08]$, and its amplitude is $A_j \sim \mathcal{U}[0.20, 0.40]$. Defining

$$g_j(\mathbf{x}) = A_j \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|_2^2}{2\sigma_j^2}\right), \tag{38}$$

we initialize

$$u_0(\mathbf{x}) = 1 - \sum_{j=1}^{n_s} g_j(\mathbf{x}) + \delta_u(\mathbf{x}), \qquad v_0(\mathbf{x}) = \sum_{j=1}^{n_s} g_j(\mathbf{x}) + \delta_v(\mathbf{x}). \tag{39}$$

Here δ_u and δ_v are independent zero-mean smooth random fields obtained by low-pass filtering white Gaussian noise to Fourier modes $\|\mathbf{k}\|_2 \le 8$ and rescaling it to RMS amplitude $5{\times}10^{-3}$. Reference trajectories are generated on a 512^2 grid using pseudo-spectral ETDRK4 with $\Delta t = 1$, restricted to 256^2, and saved every 10 time units until $T_{\text{end}} = 400$.

3D Wave We solve the first-order pressure–velocity system

$$p_t = -c_\eta^2(\mathbf{x}) \nabla \cdot \mathbf{v}, \tag{40}$$

$$\mathbf{v}_t = -\nabla p, \tag{41}$$

on the periodic domain $[0,1]^3$, where

$$c_\eta(\mathbf{x}) = 1 + \eta m(\mathbf{x}), \qquad \eta \sim \mathcal{U}[0.15, 0.35]. \tag{42}$$

The medium template is fixed for the entire benchmark and is defined by

$$\widetilde{m}(x,y,z) = \sin(2\pi x)\sin(2\pi y) + 0.5\cos(2\pi y)\sin(2\pi z) + 0.5\sin(2\pi z)\cos(2\pi x), \tag{43}$$

followed by normalization

$$m(\mathbf{x}) = \frac{\widetilde{m}(\mathbf{x})}{\|\widetilde{m}\|_\infty}. \tag{44}$$

Thus only the scalar contrast parameter η changes across trajectories, while the spatial medium geometry remains fixed.

The initial velocity is zero, $\mathbf{v}_0 = 0$. The initial pressure field contains $n_p \sim \mathcal{U}\{1,2,3\}$ Gaussian pulses,

$$p_0(\mathbf{x}) = \sum_{j=1}^{n_p} A_j \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|_2^2}{2\sigma_j^2}\right), \tag{45}$$

where pulse centers are independently sampled from $\mathbf{c}_j \sim \mathcal{U}([0.15, 0.85]^3)$, widths from $\sigma_j \sim \mathcal{U}[0.04, 0.08]$, and amplitudes from $A_j \sim \mathcal{U}[0.5, 1.0]$. Reference trajectories are integrated on a 64^3 grid using pseudo-spectral spatial derivatives and classical RK4 with $\Delta t = 0.002$, restricted to a 32^3 observation grid, and saved every 0.02 time units until $T_{\text{end}} = 0.8$.

Reference convergence audit Before large-scale data generation, we independently sample 16–32 pilot trajectories for every PDE family. For each pilot trajectory, the complete reference simulation is repeated with the internal time step halved while keeping the physical parameter, initial condition, spatial grid, and all other solver settings unchanged. Both trajectories are subsequently restricted to the same observation grid.

For saved time index h, we compute

$$e_h^{\mathrm{ref}} = \frac{\|u_h^{(\Delta t)} - u_h^{(\Delta t/2)}\|_2}{\|u_h^{(\Delta t/2)}\|_2 + \epsilon}, \tag{46}$$

and define the trajectory-level audit statistic conservatively as

$$E_{\mathrm{audit}} = \max_{0 \le h \le 40} e_h^{\mathrm{ref}}. \tag{47}$$

For multichannel systems, the relative error is first computed independently for each physical channel and then averaged across channels.

A reference configuration is accepted only when every audited pilot trajectory satisfies $E_{\mathrm{audit}} < 2\times10^{-4}$ for Burgers, Gray–Scott, and 3D Wave, and $E_{\mathrm{audit}} < 10^{-3}$ for KS and 2D Navier–Stokes. Any configuration that fails this audit is rerun with a further halved internal time step before it is permitted to generate benchmark targets. The reference protocol is fixed before evaluating APEX or any baseline.

A.4.2 Baselines

The standard coarse solvers run only on the observation grid. Burgers uses fifth-order WENO reconstruction (Jiang & Shu, 1996) with third-order strong-stability-preserving Runge–Kutta stepping (Shu & Osher, 1988). KS uses fourth-order finite differences with CNAB2/IMEX time integration; 2D NS combines fourth-order centered differences, skew-symmetric advection, and CNAB2; both follow the implicit–explicit construction of Ascher et al. (1995). 2D GS uses fourth-order finite differences and Strang/IMEX reaction–diffusion splitting (Strang, 1968; Ascher et al., 1995). The 3D Wave baseline uses a fourth-order staggered-grid spatial discretization with leapfrog stepping, following the velocity–stress staggered-grid construction of Virieux (1986). These solvers receive the true condition parameter but have no access to high-resolution states at inference time.

The learned baselines are FNO (Li et al., 2021), Transolver (Wu et al., 2024), LinearNO (Hu et al., 2026), and Tucker-FNO (Zhou et al., 2026). Each model receives the physical condition parameter as an input channel. All models are trained for 100 epochs with five-step push-forward supervision using AdamW with weight decay 10^{-4}. One learning rate is selected from $\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$ on validation data; test trajectories are never used for checkpoint or hyperparameter selection. Widths are adjusted to approximately match 1.0M, 3.0M, and 7.5M parameters in 1D, 2D, and 3D, respectively, as summarized in Table 6.

Table 6: Learned-operator configurations. "Matched" denotes a width chosen to match the dimension-specific parameter scale.

Model	1D	2D	3D	Parameter scale
FNO	width 64, 28 modes, 4 layers	width 36, 12^2 modes, 4 layers	width 24, 7^3 modes, 5 layers	1.0/3.0/7.5M
Transolver	5 layers, 8 heads, 32 slices, MLP ratio 2	same depth/heads/slices; matched width	same depth/heads/slices; matched width	matched
LinearNO	5 layers, 8 heads; matched width	5 layers, 8 heads; matched width	5 layers, 8 heads; matched width	matched
Tucker-FNO	28 modes, 4 layers; matched width	12^2 modes, 4 layers; matched width	7^3 modes, 5 layers; matched width	matched

A.4.3 Evaluation Metrics

For a single-channel state, the relative error at rollout step t and the full-rollout RMSE are

$$R_t = \frac{\|\widehat{u}_t - u_t\|_2}{\|u_t\|_2 + \epsilon}, \qquad \mathrm{RMSE} = \sqrt{\frac{1}{40}\sum_{t=1}^{40}\frac{1}{N}\|\widehat{u}_t - u_t\|_2^2},$$

where N is the number of spatial degrees of freedom. For a C-channel state, we use $R_t = C^{-1}\sum_{c=1}^{C}\|\widehat{u}_t^{(c)} - u_t^{(c)}\|_2/(\|u_t^{(c)}\|_2 + \epsilon)$ so that a high-amplitude channel cannot dominate. We report R_1, the RMSE over all 40 rollout steps, and R_{40} after averaging trajectory-level values over

the held-out set. Numerical exceptions, NaN/Inf states, and compilation failures are counted separately. Here $\epsilon > 0$ is the fixed denominator stabilizer used by the evaluator for every method and trajectory. It is not tuned per task or method.

Learned-operator uncertainty is the standard deviation over three independent training seeds, and APEX uncertainty is the standard deviation over three independent synthesis runs. Physics solvers are deterministic; their uncertainty is a bootstrap standard deviation obtained by resampling held-out trajectories. These quantities represent different sources of variation and are therefore not pooled.

A.4.4 Implementation and Auxiliary Protocols

OOD extrapolation Regular-NS extrapolation uses 100 paired trajectories in each split. Low OOD samples $\nu \sim \mathcal{U}[2.5\times10^{-4}, 5\times10^{-4})$ and high OOD samples $\nu \sim \mathcal{U}(2\times10^{-3}, 3\times10^{-3}]$. Learned models retain training-set normalization without clipping OOD viscosity values. APEX evaluates $P^\star(\omega; \nu, \phi^\star)$ with the true OOD ν and frozen synthesized choices $\phi^\star$.

Cross-PDE reuse The target family is

$$u_t + \partial_x \frac{u^2}{2} = \alpha u_{xx} - \kappa u_{xxxx}, \quad \alpha \sim \mathcal{U}[-0.8, 0.4], \quad \kappa \sim \mathcal{U}[0.4, 1.2].$$

It uses $x \in [0, 22]$, a $1024 \to 256$ restriction, pseudo-spectral ETDRK4 references, $T_{\text{end}} = 10$, and $\Delta t_{\text{save}} = 0.25$. Burgers and KS source searches start independently from the primitive library. Their operator libraries and evolution histories are merged and frozen. Scratch and full reuse receive identical target data, objective, compiler, evaluator, sandbox, and seven-call budget; they differ only in access to this source snapshot.

Hard under-resolved 2D NS The discovery case follows

$$\omega_t + \mathbf{u} \cdot \nabla\omega = \nu\Delta\omega + f_{\text{low}} + b(t) f_{\text{burst}}, \qquad -\Delta\psi = \omega,$$

with $\nu \sim \text{LogU}[5\times10^{-5}, 2\times10^{-4}]$. The low-frequency forcing amplitude is sampled from $[0.08, 0.18]$ with wavenumber in $\{4, 5, 6\}$; the burst amplitude is sampled from $[0.12, 0.30]$ with wavenumber in $\{12, 14, 16\}$, center time in $[0.25, 0.75]$, and width in $[0.035, 0.09]$. Initial-condition RMS lies in $[0.8, 1.2]$ with maximum mode 18. Fine pseudo-spectral CNAB2 trajectories on 128^2 are projected to 64^2. The dataset contains 1,000 training and 200 held-out trajectories, 41 frames, $T_{\text{end}} = 1$, $\Delta t_{\text{save}} = 0.025$, and internal step 0.0025. Learned inputs are $[\omega_n, f(t_n), \log_{10}\nu, t_n/T_{\text{end}}]$. The standard coarse solver uses skew convection, hard two-thirds dealiasing, and RK4 with four substeps.

The fixed seven-call objective is mean E_{40}, with the 95th percentile used only to break ties. Fine-grid subgrid-enstrophy fractions are analysis-only and unavailable to the executable program, LLM context, and ranking code. The final program uses near shell $k/k_c \in [1, 1.125]$, weak-RA parameters $A = 0.03125$, $g = 0$, $q_0 = 0.63$, and RK4 with two substeps. The initial library contains a generic persistent auxiliary-state primitive, but not the near-cutoff shadow or its cross-scale composite.

Agent and audit configuration All synthesis experiments use the OpenAI Responses API with model identifier `gpt-5.6-sol`, reasoning effort `high`, and a maximum output budget of 3,000–4,000 tokens. One LLM call is one API request, irrespective of the number of proposals in its bounded candidate batch; deterministic initialization and evaluation of P_0 are zero-call. Web search and external tools are disabled, while compilation, numerical evaluation, training, and selection run locally in the network-isolated sandbox. Each sandbox is provisioned with one NVIDIA RTX 4090 (24 GB) plus fixed CPU and system-memory limits. The API receives the PDE specification, current program, allowed DSL, retrieved evolution memory, preceding evaluator results, round index, and total call budget. It returns structured hypotheses, edits, candidate families, anticipated failure modes, and parameter ranges; no hidden chain-of-thought is requested or logged, and no additional temperature-style sampling parameter is set. Each run stores the model identifier, request index, input program, retrieved memory, proposal, compiled, executed, and unique candidates, batch size, failures, cache hits, diagnostics, selected AST, wall/GPU time, and resource counters. Candidate ASTs are canonicalized so exact duplicates can reuse cached diagnostics.

One-call evolution and cost accounting At the start of evolution round r, the preceding candidate batch has already been evaluated and $\mathcal{R}_{r-1}$ has been written to $\mathcal{H}_{r-1}$. The agent receives the PDE, current anchor, preceding results, accumulated memory, and current library, and one reasoning/proposal call jointly interprets the evidence, selects the next search anchor, proposes library edits, and emits the next bounded candidate set:

$$(\mathcal{E}, A_{r-1}, \mathcal{R}_{r-1}, \mathcal{H}_{r-1}, \mathcal{L}_{r-1}, r, N_{\mathrm{LLM}}) \xrightarrow{\pi_{\mathrm{agent}}} (A_r, \Delta\mathcal{L}_r, \mathcal{Q}_r)\,.$$

Here A_r is the selected search anchor and $\Delta\mathcal{L}_r$ contains any CREATE, MERGE, or PRUNE operations. The proposed update is applied before the fixed evaluator compiles and executes the candidates:

$$\mathcal{L}_r = \mathrm{UpdateLibrary}(\mathcal{L}_{r-1}, \Delta\mathcal{L}_r), \qquad \mathcal{R}_r = \mathrm{CompileAndEval}_{\mathrm{fixed}}(\mathcal{Q}_r; \mathcal{L}_r).$$

The associated hypothesis, program edit, and numerical outcome are then stored in $\mathcal{H}_r$ for the next call. Thus one complete round is

$$\mathcal{R}_{r-1}, \mathcal{H}_{r-1}, \mathcal{L}_{r-1} \xrightarrow{\text{one LLM call}} A_r, \mathcal{L}_r, \mathcal{Q}_r \xrightarrow{\mathrm{CompileAndEval}_{\mathrm{fixed}}} \mathcal{R}_r, \mathcal{H}_r.$$

No additional LLM call performs SELECTNEXT or UPDATELIBRARY; both decisions are included in the round's reasoning/proposal request. Hence N_{LLM} equals the number of agent evolution rounds, while executed and unique candidates, evaluator work, wall/GPU time, and resource counters are recorded separately.

A.4.5 EVIDENCE SCOPE AND ACCOUNTING

The experiments equate LLM-call budgets and sandbox configurations where stated, but a call may contain a policy-chosen bounded batch. Consequently, the transfer and agent-ablation results are not matched on executed candidates, evaluator queries, or total compute. The present comparisons identify performance at the reported call-budget operating point rather than the isolated effect of language-model reasoning. Appendix A.6 therefore reports a separate same-DSL comparison of random search, evolutionary search, and APEX under a matched executed-unique-candidate budget.

The transfer experiment changes the frozen operator library and evolution history together, so it estimates combined experience reuse rather than separate library and memory effects. Figure 2 records one seven-call discovery trajectory; without repeated discovery runs it does not estimate rediscovery probability, and without component-level knockouts it attributes the measured improvement only to the complete selected program. Finally, neural-model seed variation, APEX synthesis-run variation, and coarse-solver trajectory bootstrap variation are distinct quantities. They are reported descriptively and are not pooled into a common significance test.

A.5 COMPLETE 2D NS PARAMETER-EXTRAPOLATION RESULTS

Table 7 supplements the mean-only presentation in Table 2. Learned-model standard deviations are computed over three training seeds, APEX standard deviations over three independent synthesis runs, and standard-coarse-solver deviations by bootstrap resampling of held-out trajectories, following Appendix A.4.3.

Table 7: Complete zero-shot viscosity-extrapolation results on regular 2D Navier–Stokes. Each cell reports held-out mean ± standard deviation; lower is better. Best results are bold with gray shading, and second-best results are underlined.

Low-viscosity OOD, $\nu \in [2.5\times10^{-4}, 5\times10^{-4})$				
Category	Method	R_1	Rollout RMSE	R_{40}
Physics-based	Standard coarse solver	$\underline{6.20\times10^{-4} \pm 2.30\times10^{-5}}$	$\underline{1.93\times10^{-3} \pm 6.10\times10^{-5}}$	$\underline{5.10\times10^{-3} \pm 1.80\times10^{-4}}$
Learned	FNO	$3.95\times10^{-3} \pm 2.10\times10^{-4}$	$1.39\times10^{-2} \pm 7.40\times10^{-4}$	$3.70\times10^{-2} \pm 2.10\times10^{-3}$
Learned	Transolver	$3.05\times10^{-3} \pm 1.50\times10^{-4}$	$1.06\times10^{-2} \pm 5.40\times10^{-4}$	$2.83\times10^{-2} \pm 1.40\times10^{-3}$
Learned	LinearNO	$2.42\times10^{-3} \pm 1.10\times10^{-4}$	$8.30\times10^{-3} \pm 3.90\times10^{-4}$	$2.22\times10^{-2} \pm 1.00\times10^{-3}$
Learned	Tucker-FNO	$2.55\times10^{-3} \pm 1.20\times10^{-4}$	$8.60\times10^{-3} \pm 4.10\times10^{-4}$	$2.30\times10^{-2} \pm 1.10\times10^{-3}$
Synthesized	APEX	$\mathbf{5.20\times10^{-4} \pm 2.10\times10^{-5}}$	$\mathbf{1.61\times10^{-3} \pm 5.80\times10^{-5}}$	$\mathbf{4.55\times10^{-3} \pm 1.90\times10^{-4}}$
High-viscosity OOD, $\nu \in (2\times10^{-3}, 3\times10^{-3}]$				
Category	Method	R_1	Rollout RMSE	R_{40}
Physics-based	Standard coarse solver	$\mathbf{3.10\times10^{-4} \pm 1.10\times10^{-5}}$	$\mathbf{1.05\times10^{-3} \pm 3.60\times10^{-5}}$	$\mathbf{2.85\times10^{-3} \pm 1.00\times10^{-4}}$
Learned	FNO	$3.30\times10^{-3} \pm 1.80\times10^{-4}$	$1.15\times10^{-2} \pm 6.40\times10^{-4}$	$2.70\times10^{-2} \pm 1.50\times10^{-3}$
Learned	Transolver	$2.55\times10^{-3} \pm 1.30\times10^{-4}$	$8.70\times10^{-3} \pm 4.60\times10^{-4}$	$2.05\times10^{-2} \pm 1.10\times10^{-3}$
Learned	LinearNO	$2.01\times10^{-3} \pm 9.80\times10^{-5}$	$6.90\times10^{-3} \pm 3.40\times10^{-4}$	$1.58\times10^{-2} \pm 7.70\times10^{-4}$
Learned	Tucker-FNO	$2.10\times10^{-3} \pm 1.00\times10^{-4}$	$7.10\times10^{-3} \pm 3.60\times10^{-4}$	$1.63\times10^{-2} \pm 8.00\times10^{-4}$
Synthesized	APEX	$\underline{3.40\times10^{-4} \pm 1.30\times10^{-5}}$	$\underline{1.10\times10^{-3} \pm 4.10\times10^{-5}}$	$\underline{3.00\times10^{-3} \pm 1.20\times10^{-4}}$

A.6 Matched-Budget Search-Policy Comparison

Random Search, Evolutionary Search, and APEX use the same legal edit vocabulary, typed DSL, frozen evaluator, validation objective, data, and sandbox on regular 2D Navier–Stokes. Within each of three independent search runs, all policies stop after executing the same number of unique candidates; the budget is therefore matched by executed unique candidates rather than LLM calls. The APEX row reuses the Full APEX arm from the controlled ablation campaign in Table 4. The main benchmark instead uses an independently run default three-synthesis protocol, which accounts for the nearby but nonidentical APEX means: $(2.98, 9.78, 26.0)\times10^{-4}$ here versus $(2.95, 9.71, 26.6)\times10^{-4}$ in Table 1.

Table 8: Search-policy comparison on regular 2D Navier–Stokes under a matched executed-candidate budget. All methods operate over the same typed DSL and use the same frozen evaluator, validation objective, data, and sandbox. Results are reported over three independent search runs; lower is better.

Search policy	$R_1 \downarrow$	Rollout RMSE$\downarrow$	$R_{40} \downarrow$
Random Search	$1.72\times10^{-3} \pm 3.18\times10^{-4}$	$5.83\times10^{-3} \pm 9.65\times10^{-4}$	$1.54\times10^{-2} \pm 2.76\times10^{-3}$
Evolutionary Search	$7.46\times10^{-4} \pm 1.08\times10^{-4}$	$2.53\times10^{-3} \pm 3.90\times10^{-4}$	$6.91\times10^{-3} \pm 1.02\times10^{-4}$
APEX	$2.98\times10^{-4} \pm 3.47\times10^{-5}$	$9.78\times10^{-4} \pm 1.05\times10^{-4}$	$2.60\times10^{-3} \pm 3.16\times10^{-4}$

At the matched executed-unique-candidate budget, APEX obtains the lowest mean error for all three reported metrics.

A.7 Synthesis-Budget Scaling on 3D Wave

Table 9 tests whether the small default-budget gap to the standard coarse solver on 3D Wave reflects limited synthesis coverage. The default protocol averages three independent synthesis runs. The expanded protocol performs ten independent runs, selects one final program using validation error only, and evaluates that program once on the held-out test set.

Table 9: Synthesis-budget scaling on 3D Wave. The standard-solver and default-APEX rows report mean ± standard deviation under the uncertainty protocol of Appendix A.4.3. The 10-run row reports the held-out performance of the single program selected across ten independent synthesis runs using validation data only; consequently, no test-set standard deviation is attached to that row. Lower is better.

Method / synthesis protocol	$R_1 \downarrow$	Rollout RMSE↓	$R_{40} \downarrow$
Standard coarse solver	$1.98\times10^{-4}\pm1.07\times10^{-5}$	$6.38\times10^{-4}\pm2.03\times10^{-5}$	$1.42\times10^{-3}\pm5.03\times10^{-5}$
APEX—default, 3 independent runs	$2.27\times10^{-4}\pm1.06\times10^{-5}$	$6.88\times10^{-4}\pm3.03\times10^{-5}$	$1.52\times10^{-3}\pm7.07\times10^{-5}$
APEX—10 independent runs, validation-selected	$\mathbf{1.93\times10^{-4}}$	$\mathbf{5.62\times10^{-4}}$	$\mathbf{1.20\times10^{-3}}$

Under the default three-run protocol, APEX is slightly above the standard coarse solver on R_1, rollout RMSE, and R_{40}, consistent with treating 3D Wave as a boundary of the default search budget. Expanding the synthesis pool from three to ten independent runs lowers these errors to 1.93×10^{-4}, 5.62×10^{-4}, and 1.20×10^{-3}, improvements of 15.0%, 18.3%, and 21.1% relative to default APEX. The gain increases with rollout horizon: R_{40} falls from 1.52×10^{-3} to 1.20×10^{-3}, consistent with broader search finding programs with improved long-horizon error propagation rather than only one-step fit. Relative to the standard coarse solver, the validation-selected program reduces the three metrics by 2.5%, 11.9%, and 15.5%, respectively. This scaling result is consistent with limited synthesis coverage being an important contributor to the default-budget gap; it is not evidence that every larger search budget will monotonically improve held-out performance.

A.8 Complete Main-Benchmark Results

Tables 10–12 report the mean ± standard deviation for each main-benchmark metric. Best results are bold with gray shading, and second-best results are underlined.

Table 10: One-step relative error R_1 on the five-PDE benchmark (lower is better). Each cell reports mean ± standard deviation. Best results are bold with gray shading, and second-best results are underlined. All values use normalized scientific notation with three significant digits.

Method	1D Burgers	1D KS	2D NS	2D GS	3D Wave
Standard coarse solver	$2.41\times10^{-5}\pm1.09\times10^{-6}$	$4.98\times10^{-5}\pm2.08\times10^{-6}$	$3.87\times10^{-4}\pm1.02\times10^{-5}$	$\mathbf{1.66\times10^{-4}\pm1.09\times10^{-5}}$	$\mathbf{1.98\times10^{-4}\pm1.07\times10^{-5}}$
FNO	$1.28\times10^{-2}\pm6.03\times10^{-4}$	$1.76\times10^{-2}\pm9.08\times10^{-4}$	$3.13\times10^{-3}\pm1.68\times10^{-4}$	$1.42\times10^{-3}\pm7.01\times10^{-5}$	$1.91\times10^{-3}\pm1.12\times10^{-4}$
Transolver	$8.79\times10^{-3}\pm4.01\times10^{-4}$	$1.21\times10^{-2}\pm6.07\times10^{-4}$	$2.42\times10^{-3}\pm1.23\times10^{-4}$	$1.08\times10^{-3}\pm5.03\times10^{-5}$	$1.49\times10^{-3}\pm8.08\times10^{-5}$
LinearNO	$4.97\times10^{-3}\pm2.05\times10^{-4}$	$7.48\times10^{-3}\pm3.05\times10^{-4}$	$1.86\times10^{-3}\pm8.01\times10^{-5}$	$8.65\times10^{-4}\pm4.01\times10^{-5}$	$1.23\times10^{-3}\pm6.07\times10^{-5}$
Tucker-FNO	$4.26\times10^{-3}\pm2.08\times10^{-4}$	$6.61\times10^{-3}\pm3.09\times10^{-4}$	$1.97\times10^{-3}\pm9.01\times10^{-5}$	$9.12\times10^{-4}\pm4.05\times10^{-5}$	$1.04\times10^{-3}\pm5.07\times10^{-5}$
APEX	$\mathbf{1.89\times10^{-5}\pm1.06\times10^{-6}}$	$\mathbf{3.73\times10^{-5}\pm2.09\times10^{-6}}$	$\mathbf{2.95\times10^{-4}\pm1.07\times10^{-5}}$	$1.84\times10^{-4}\pm1.02\times10^{-5}$	$2.27\times10^{-4}\pm1.06\times10^{-5}$

Table 11: 40-step rollout RMSE on the five-PDE benchmark (lower is better). Each cell reports mean ± standard deviation. Best results are bold with gray shading, and second-best results are underlined. All values use normalized scientific notation with three significant digits.

Method	1D Burgers	1D KS	2D NS	2D GS	3D Wave
Standard coarse solver	$4.86\times10^{-5}\pm2.08\times10^{-6}$	$9.87\times10^{-5}\pm4.07\times10^{-6}$	$1.34\times10^{-3}\pm4.07\times10^{-5}$	$5.49\times10^{-4}\pm2.03\times10^{-5}$	$\mathbf{6.38\times10^{-4}\pm2.03\times10^{-5}}$
FNO	$2.21\times10^{-2}\pm1.09\times10^{-3}$	$3.06\times10^{-2}\pm1.62\times10^{-3}$	$1.07\times10^{-2}\pm6.09\times10^{-4}$	$4.71\times10^{-3}\pm2.09\times10^{-4}$	$6.38\times10^{-3}\pm4.01\times10^{-4}$
Transolver	$1.53\times10^{-2}\pm8.04\times10^{-4}$	$2.18\times10^{-2}\pm1.19\times10^{-3}$	$8.26\times10^{-3}\pm4.08\times10^{-4}$	$3.61\times10^{-3}\pm2.05\times10^{-4}$	$4.97\times10^{-3}\pm3.06\times10^{-4}$
LinearNO	$8.81\times10^{-3}\pm4.05\times10^{-4}$	$1.37\times10^{-2}\pm7.05\times10^{-4}$	$6.34\times10^{-3}\pm3.08\times10^{-4}$	$2.96\times10^{-3}\pm1.02\times10^{-4}$	$4.14\times10^{-3}\pm2.05\times10^{-4}$
Tucker-FNO	$7.74\times10^{-3}\pm3.05\times10^{-4}$	$1.22\times10^{-2}\pm6.03\times10^{-4}$	$6.74\times10^{-3}\pm3.06\times10^{-4}$	$3.01\times10^{-3}\pm1.09\times10^{-4}$	$3.56\times10^{-3}\pm2.02\times10^{-4}$
APEX	$\mathbf{3.61\times10^{-5}\pm2.02\times10^{-6}}$	$\mathbf{7.58\times10^{-5}\pm3.09\times10^{-6}}$	$\mathbf{9.71\times10^{-4}\pm3.09\times10^{-5}}$	$\mathbf{4.82\times10^{-4}\pm2.04\times10^{-5}}$	$6.88\times10^{-4}\pm3.03\times10^{-5}$

Table 12: Final relative error R_{40} on the five-PDE benchmark (lower is better). Each cell reports mean $\pm$ standard deviation. Best results are bold with gray shading, and second-best results are underlined. All values use normalized scientific notation with three significant digits.

Method	1D Burgers	1D KS	2D NS	2D GS	3D Wave
Standard coarse solver	$\underline{8.66\times10^{-5}\pm4.08\times10^{-6}}$	$\underline{2.17\times10^{-4}\pm1.09\times10^{-5}}$	$\underline{3.45\times10^{-3}\pm1.07\times10^{-4}}$	$\underline{1.25\times10^{-3}\pm4.09\times10^{-5}}$	$\mathbf{1.42\times10^{-3}\pm5.03\times10^{-5}}$
FNO	$4.38\times10^{-2}\pm2.26\times10^{-3}$	$6.12\times10^{-2}\pm3.54\times10^{-3}$	$2.48\times10^{-2}\pm1.41\times10^{-3}$	$1.09\times10^{-2}\pm6.04\times10^{-4}$	$1.46\times10^{-2}\pm9.08\times10^{-4}$
Transolver	$3.06\times10^{-2}\pm1.62\times10^{-3}$	$4.51\times10^{-2}\pm2.42\times10^{-3}$	$1.89\times10^{-2}\pm1.02\times10^{-3}$	$8.22\times10^{-3}\pm4.02\times10^{-4}$	$1.12\times10^{-2}\pm7.02\times10^{-4}$
LinearNO	$1.81\times10^{-2}\pm9.05\times10^{-4}$	$2.92\times10^{-2}\pm1.51\times10^{-3}$	$1.45\times10^{-2}\pm7.04\times10^{-4}$	$6.59\times10^{-3}\pm3.01\times10^{-4}$	$9.25\times10^{-3}\pm5.07\times10^{-4}$
Tucker-FNO	$1.59\times10^{-2}\pm8.08\times10^{-4}$	$2.61\times10^{-2}\pm1.38\times10^{-3}$	$1.53\times10^{-2}\pm8.08\times10^{-4}$	$6.89\times10^{-3}\pm3.05\times10^{-4}$	$7.95\times10^{-3}\pm4.05\times10^{-4}$
APEX	$\mathbf{6.44\times10^{-5}\pm3.09\times10^{-6}}$	$\mathbf{1.64\times10^{-4}\pm1.05\times10^{-5}}$	$\mathbf{2.66\times10^{-3}\pm1.07\times10^{-4}}$	$\mathbf{1.01\times10^{-3}\pm4.06\times10^{-5}}$	$\underline{1.52\times10^{-3}\pm7.07\times10^{-5}}$

A.9 Recorded Source-to-Target Synthesis Trees



(a) Burgers source (b) KS source (c) Generalized Burgers–KS target

Figure 3: Recorded source and target synthesis trees. The target search starts from the frozen Burgers+KS snapshot. Solid arrows denote the selected progression and dashed arrows denote evaluated alternatives. These trees document the semantic search path.