SPADES: A Stochastic Physics-Guided Adaptive Denoising Equilibration Solver for Discretized PDEs

A Diffusion-Based Gibbs Formulation of Finite-Difference and Finite-Element Solvers

Park Junhu
nller.park@snu.ac.kr

August 6, 2026

Manuscript Status

This document is a **non-peer-reviewed preprint draft** with reproducible theory, code, and numerical experiments. It does not claim to introduce physics-informed diffusion itself. Its provisional contributions are to define a Gibbs measure from discrete FDM/FEM residuals, state a deterministic low-temperature limit and a mesh–temperature coupling rate, connect mean–fluctuation factorization and residual whitening to learning design for linear SPD problems, implement a physical-initializer residual diffusion for a variable-coefficient Poisson pilot, and add a seed-disjoint random-map benchmark for reusable solver-error learning. The random maps are not the final scientific object; they are a controlled way to generate many different boundary-value problems for the same fixed lightweight FDM solver. The benchmark trains eight independent SPADES models and matched deterministic regressors on 48 random internal-boundary maps per run, selects checkpoints by seed-disjoint validation maps, and evaluates 48 unseen maps per test regime and seed. It provides empirical evidence that a learned solver-error law can transfer to previously unseen physical configurations under the implemented generator and fixed solver. It is not peer-reviewed and is not a universal claim about arbitrary geometries, Navier–Stokes, FEM/FVM transfer, or calibrated uncertainty.

Abstract

Classical finite-difference methods (FDM) and finite-element methods (FEM) compute one deterministic vector satisfying a prescribed discrete system. This work lifts that computation to a probability-measure problem on the discrete solution space. Given a discrete residual $R_h(z;c)$ and a symmetric positive-definite metric W_h, we define the energy

$$E_h(z;c) = \tfrac{1}{2} R_h(z;c)^{\mathsf{T}} W_h R_h(z;c) + \Phi_h(z;c)$$

and take the Gibbs law $\pi_{\beta,h}(\mathrm{d}z \mid c) \propto \exp[-\beta E_h(z;c)]\mathrm{d}z$ at inverse temperature β as the "stochastic discrete solution." If the energy is coercive, the measure is well defined and concentrates on the set of classical discrete solutions as $\beta \to \infty$. For linear SPD FEM, this law is exactly $\mathcal{N}(K^{-1}b, \beta^{-1}K^{-1})$, and when $K = LL^{\mathsf{T}}$ the whitened residual $y = \sqrt{\beta}L^{-1}(Kz - b)$ is standard normal. This structure yields (i) the temperature separation $z = z_h^* + \beta^{-1/2}v$, (ii) the equipartition identity $\mathbb{E}\,\|z - z_h^*\|_K^2 = n_h/\beta$, and (iii) a mesh–temperature condition $\beta_h \gtrsim n_h h^{-2p}$ that preserves the FEM convergence order.

The learning method separates a deterministic mean operator from a temperature-independent fluctuation diffusion. The mean is trained with supervised and Galerkin-residual losses; the fluctuation model is trained with denoising score matching, whitened-residual moment matching, and, in the linear verification problem, analytic Gaussian score distillation. The proposed model is not a neural replacement for FDM or FEM. It is a stochastic correction layer inside an existing numerical pipeline: a physics initializer is mapped to a residual or correction space, conditionally denoised, corrected by a discrete operator step, and projected back to the constraint set. Relative to a conventional deterministic PINN workflow, the possible advantage is problem-dependent and comes from amortizing this fixed correction pipeline over many related queries and from representing a conditional law over discrete fields. We prove only a conditional complexity statement, not universal superiority over PINNs, neural operators, or classical solvers. We also prove the ideal convergence of exact reverse denoising, strongly convex Gibbs relaxation, and SPD solver-native correctors under explicit assumptions. On a one-dimensional Poisson P1-FEM experiment with 32 degrees of freedom, the mean operator reaches relative error 1.28×10^{-5}, the

equipartition ratio is 1.037, the score-matrix relative error is 0.0071, and the temperature-scaling slope is −0.500. In a variable-coefficient Poisson pilot, a 10-node coarse FDM initializer is corrected by residual diffusion, reducing mean relative L^2 error from 0.0272 to 0.0191 in the operator-corrected hybrid, but at much higher CPU cost than a direct Jacobi correction baseline. In a new random-map benchmark, eight independently trained models each use 48 training maps and 48 seed-disjoint validation maps. The maps generate distinct boundary-value problems for the same fixed FDM solver, so unseen maps operationalize unseen physical configurations. On 48 unseen maps per test regime and seed, SPADES improves the fixed-FDM initializer across IID, complexity-OOD, topology-OOD, narrow-channel-OOD, and joint-OOD regimes. Its mean correction gain exceeds a matched deterministic regressor by 0.019–0.025 across the five regimes. The advantage is modest but consistent under a small matched 133k-parameter backbone, where SPADES also carries the extra representational burden of timestep-conditioned denoising. These experiments support the view that the learned object is a reusable numerical solver-error law rather than an individual solution field, while the theoretical reason for the modest diffusion advantage remains open.

Keywords: probabilistic numerics, finite element method, finite difference method, Gibbs measure, diffusion model, score matching, uncertainty quantification

Contents

1 Introduction **4**

2 Related Work and Positioning **5**

2.1 Probabilistic numerics and probabilistic FEM 5

2.2 Physics-informed neural solvers 5

2.3 Neural operators and geometry-aware surrogates 6

2.4 Solver-in-the-loop learning, learned discretization, and correction 6

2.5 Diffusion-based PDE models 7

3 Amortized Complexity Relative to Per-Instance PINN Optimization **8**

4 Discrete Gibbs Formulation **9**

4.1 General residual-based definition 9

4.2 Exact distribution for linear residuals 9

4.3 SPD FEM variational Gibbs law 10

5 Mesh–Temperature Coupling and Continuum Limit **10**

6 Convergence Guarantees **11**

7 Diffusion and Score Learning **13**

7.1 Gibbs score and stochastic relaxation 13

7.2 Physical-initializer multi-condition residual diffusion 13

7.3 Condition-wise reverse-start selection from the initial residual 15

7.4 Why direct coefficient diffusion is unstable 17

7.5 Mean–fluctuation factorization 17

7.6 Whitened-residual moment loss 17

7.7 Linear Gaussian score distillation 18

8 Algorithms **19**

9 One-Dimensional Poisson FEM Verification **19**
9.1 Problem and data . . . 19
9.2 Model and training setup . . . 20
9.3 Evaluation metrics . . . 20
9.4 Results . . . 20
9.5 Interpretation . . . 21

10 Variable-Coefficient Poisson Warm-Start Pilot **21**
10.1 Problem, inputs, and initializer . . . 21
10.2 Network and training . . . 23
10.3 Pilot results . . . 24

11 Random-Geometry Generalization from Seed-Disjoint Internal-Boundary Maps **25**
11.1 Scientific objective . . . 25
11.2 Random-map generation and fixed solvers . . . 25
11.3 Seed-disjoint protocol and models . . . 26
11.4 Main results . . . 26
11.5 Interpretation and limitations of the random-geometry result . . . 27

12 Extensions **28**
12.1 Dimension-agnostic residual-correction structure . . . 28
12.2 FDM . . . 31
12.3 Nonlinear FEM . . . 31
12.4 Time-dependent PDE . . . 32
12.5 Variable meshes and function spaces . . . 32
12.6 Stochastic solver-component generation . . . 32

13 Limitations and Falsifiable Research Tasks **32**

14 Conclusion **33**

A Proof Supplements **34**
A.1 Quantitative bound for low-temperature concentration . . . 34
A.2 Cross term in the mesh–temperature identity . . . 34

B Core Implementation Excerpts **34**

C Reproduction Commands and Fixed Seeds **35**

D Notation **37**

1 Introduction

The standard output of numerical PDE analysis is a single vector. In FDM, grid values $z_h \in \mathbb{R}^{n_h}$ satisfy a difference equation $A_h z_h = b_h$; in FEM, coefficient vectors satisfy the weak form $a(u_h, v_h) = \ell(v_h)$. This computation is physically interpretable and supported by convergence theory, but it does not treat the distribution of the numerical solution itself as a primary object.

Probabilistic numerics provides a general framework for representing numerical error or unknown computational quantities as probability measures. Bayesian probabilistic numerical methods cast numerical tasks as inverse problems and formalize well-posedness, and several methods assign probability measures to differential-equation solutions [5, 6, 4]. Random-mesh FEM quantifies discretization error by randomizing the mesh of standard P1-FEM [1]. The main question in this line of work is how to honestly represent numerical uncertainty induced by limited computation.

Meanwhile, diffusion and score-based generative models create data distributions by progressive denoising [8, 20]. In PDE settings, PDE-Refiner improves high-frequency components and long rollouts through diffusion-inspired iterative refinement [14]; DiffusionPDE models joint distributions of coefficients and solutions under partial observation [9]; conditional diffusion studies address forecasting and data assimilation [19]; physics-informed diffusion injects PDE residuals into training or sampling [3, 10]; and function-space diffusion has also appeared [23, 22]. Therefore, merely using diffusion for PDEs is not the novelty of this work.

The starting question here is:

> Can the discrete residual and variational energy of FDM/FEM themselves define a probability distribution, with diffusion serving as an amortized sampler for that distribution?

We call the proposed framework *SPADES*, short for *Stochastic Physics-Guided Adaptive Denoising Equilibration Solver*. The main claim is that SPADES is a solver-coupled residual diffusion framework: it preserves the structure of finite-difference and finite-element discretizations while using a conditional generative model to represent unresolved correction fields and, when the target law is calibrated, uncertainty over physically admissible discrete solutions.

In the practical correction experiments, the predicted output is a correction field, but the reusable learned structure is the numerical solver-error law: the condition-dependent error pattern produced by one fixed lightweight solver across a family of physical configurations. Here a physical configuration may include coefficient fields, boundary conditions, forcing, geometry, initial states, observations, material parameters, and related inputs. The random maps used later in the paper vary internal boundaries while keeping other major physical parameters fixed; they are one practical benchmark mechanism for producing distinct configurations, not a defining component of SPADES.

The concrete contributions of this draft are:

1. We define a Gibbs measure from a discrete residual R_h and metric W_h, and state well-posedness and the deterministic low-temperature limit.
2. We derive the exact mean, covariance, whitened residual, and equipartition identity for linear SPD FEM.
3. We couple mesh size and inverse temperature to obtain the condition $\beta_h \gtrsim n_h h^{-2p}$ under which stochastic fluctuations do not destroy the FEM convergence order.
4. We propose a stable diffusion training method based on mean–fluctuation factorization and a residual-moment loss. Unlike plain residual minimization, which can collapse the variance to zero, the whitening moments preserve a prescribed finite temperature.
5. For a practical deterministic limit, we diffuse only the correction around a physical initializer z_h^{init}, condition on coefficient fields, forcing, boundary values, initial residuals, and coordinates, add a solver-native operator corrector, and provide a residual-adaptive rule for choosing the reverse-start level condition by condition.
6. We compare online complexity with per-instance deterministic PINN workflows: after training, SPADES uses a fixed number of denoising and correction steps, which gives an amortized advantage only under explicit O and o conditions and only for problem families where the trained correction model generalizes.

7. We provide complete Python/PyTorch reproduction code, including a distributional theorem check for one-dimensional Poisson FEM, a variable-coefficient Poisson warm-start pilot, and a seed-disjoint random-map benchmark for low-data solver-error-law learning.
8. We test whether a solver-error law learned from only 48 randomly generated physical problems transfers to seed-disjoint unseen configurations for the same fixed numerical solver. Across eight independent runs, SPADES consistently improves a fixed FDM initializer and modestly but consistently exceeds a matched deterministic correction network across five IID/OOD regimes.

2 Related Work and Positioning

2.1 Probabilistic numerics and probabilistic FEM

Bayesian probabilistic numerical methods formulate numerical tasks as conditional inference between an information operator and a quantity of interest [5]. In differential-equation settings, prior work has studied solution distributions induced by limited solver information, propagation to Bayesian inverse problems, probabilistic gradients, and mesh randomization [4, 6, 1]. SPADES belongs to this literature in that it also uses measures over numerical solutions.

The difference is the origin of the measure. Instead of first choosing a prior and observation model, this draft uses the *discrete residual or variational energy directly as the Gibbs potential*. The temperature β therefore explicitly controls the allowed deviation from the discrete physical constraint, and in linear SPD problems the covariance is determined by K^{-1}. This covariance is not aleatoric uncertainty in the physical input. It is the *solver law* defined by the chosen discrete energy and temperature. A separate hierarchical probabilistic model is needed to include model discrepancy or input uncertainty.

2.2 Physics-informed neural solvers

PINNs combine PDE residuals, boundary conditions, and observation errors in a network-output loss [18]. The mean-operator training in SPADES also uses a residual loss, but the goal is not just to fit a single function. A conventional deterministic PINN often represents a coordinate-to-solution field

$$u_\theta(x,t)$$

and minimizes a loss of the form

$$\mathcal{L}_{\mathrm{PINN}} = \mathcal{L}_{\mathrm{PDE}} + \mathcal{L}_{\mathrm{BC}} + \mathcal{L}_{\mathrm{IC}} + \mathcal{L}_{\mathrm{data}}. \tag{1}$$

In contrast, the generative target in this work is a conditional law over discretized physical fields,

$$p_\theta(u_h \mid f_h, k_h, \mathrm{BC}, \mathrm{IC}, \mathcal{M}_h)\,, \tag{2}$$

and, in the solver-coupled variant, a conditional law over corrections around a physics-based initializer,

$$u_h = u_h^{\mathrm{init}} + \delta u_h. \tag{3}$$

The full finite-temperature target is $\pi_{\beta,h}$, including its variance and correlation structure. A pure residual penalty $\mathbb{E}\,\|R_h(Z)\|^2$ can shrink all samples to the deterministic solution, so it is insufficient for representing a finite-temperature distribution.

This distinction should not be read as a universal claim against PINNs. Classical supervised correction regressors require high-fidelity correction targets. PINNs instead often try to learn from PDE constraints, boundary conditions, and sparse data without requiring paired fine-solver labels for every case. SPADES follows a different design philosophy: it starts from a physical numerical initializer and learns only the remaining correction needed by that fixed solver. Bayesian PINNs, ensemble PINNs, dropout variants, and other probabilistic neural solvers can also represent uncertainty. The difference claimed here is narrower: uncertainty is native to the conditional generative formulation rather than added through a separate Bayesian

Table 1: Careful comparison with common deterministic PINN workflows. The entries describe intended modeling roles, not universal performance rankings.

Aspect	Common deterministic PINN workflow	SPADES residual-correction workflow
Output	One optimized field or one parameterized solution map.	Samples from a conditional law over discrete fields or correction fields.
Uncertainty	Requires a Bayesian, ensemble, dropout, or related extension.	Sample mean, variance, credible regions, and branch statistics are natural outputs, but calibration must be verified.
Multiple admissible states	A single deterministic regressor can select one branch or average branches unless special mechanisms are used.	A conditional generative model can represent multiple branches when the PDE formulation, observations, forcing, coefficients, or bifurcation structure make the law multimodal.
Warm start	The neural representation is commonly optimized or trained as the solution field.	The model learns $\delta u_h = u_h^\star - u_h^{\mathrm{init}}$, which may reduce target dynamic range when the initializer is informative.
Solver compatibility	The neural network often serves as the primary residual minimizer.	Denoising can be interleaved with Jacobi, Gauss–Seidel, CG, Newton–Krylov, multigrid, domain decomposition, or FEM projection.
Mesh handling	Coordinate inputs do not automatically give mesh transfer.	Mesh transfer also requires appropriate architecture: graph message passing, finite-element coefficients, geometric multigrid features, or neural-operator structure.
Adaptive refinement	Residual weighting can be adapted, but there is no built-in mesh object in a pure coordinate network.	Sample variance, residual statistics, correction magnitude, or posterior discrepancy can serve as possible AMR indicators; this is proposed coupling, not proved here.

or ensemble construction, and the generated object is a discretization-aware field or correction field that can be passed back to a classical solver.

2.3 Neural operators and geometry-aware surrogates

Neural operators learn mappings between function spaces rather than optimizing a separate coordinate network for each PDE instance. DeepONet is based on an operator universal approximation theorem [15], and Fourier neural operators parameterize an integral-kernel operator in Fourier space for parametric PDE families [13]. Geo-FNO extends this operator-learning direction to PDEs on general geometries by learning a deformation to a latent uniform-grid domain [12]. These methods are natural points of comparison for the random-map benchmark because they also target transfer across families of PDE configurations. Unlike full-solution operator learning, SPADES uses the output of a fixed numerical solver and models the solver's remaining error. This is the practical design choice of the framework, not a claim that neural operators cannot be adapted to residual or correction learning.

2.4 Solver-in-the-loop learning, learned discretization, and correction

Several lines of work use learning to improve numerical solvers without discarding the solver. Learned discretizations optimize coarse-grid differential operators from data [2]. Solver-in-the-loop training lets a learned model interact with a differentiable iterative PDE solver during training [21]. Machine-learning-accelerated CFD inserts learned components into traditional fluid simulations while retaining numerical structure [11]. Learned multigrid and algebraic multigrid methods optimize solver components such as prolongation, restriction, or coarse-grid correction [7, 16]. SPADES is closest to this solver-coupled family in practical intent: the learned model is a correction mechanism attached to a fixed lightweight initializer,

not a standalone surrogate replacing the numerical method.

2.5 Diffusion-based PDE models

PDE-Refiner uses diffusion-inspired iterative refinement to improve high-frequency components and long rollouts [14]. DiffusionPDE learns joint distributions of solutions and coefficients for partially observed forward and inverse problems [9]; conditional diffusion studies integrate forecasting and assimilation [19]. Physics-Informed Diffusion Models include PDE residuals in diffusion training [3], and CoCoGen injects discrete physical information into a score-based generative process to improve consistency of forward and inverse samples [10]. Guided diffusion on function spaces and FunDiff extend the viewpoint to function spaces or discretization-agnostic modeling [23, 22].

The closest prior work is physics-informed diffusion and CoCoGen. Park et al. also proposed a two-stage surrogate in which a coarse S-DeepONet output is used as a physical prior and video diffusion learns only the residual [17]. Thus, the idea of conditioning residual diffusion on a coarse solution is itself prior art. The warm-start variant in this draft combines that idea with FDM/FEM discrete energy, an initial-residual channel, and a solver-native correction at each reverse step. Zhang, Wang, and Zou noted that a Jensen gap can arise when PDE constraints are imposed directly on predicted clean means at noisy levels [24]. Accordingly, this draft explicitly distinguishes the residual-zero loss used for deterministic solver pilots from the residual-distribution loss required by a finite-temperature Gibbs law.

The contribution should therefore not be stated as the first PDE diffusion method. The strongest practical distinction is the solver-coupled residual formulation:

$$\begin{aligned}&\text{physics initializer} \to \text{residual or correction space} \to \text{conditional denoising}\\ &\to \text{discrete operator corrector} \to \text{constraint projection.}\end{aligned}$$

Common generative PDE approaches may construct a field from noise or denoise a globally corrupted field. SPADES starts from an existing numerical approximation and learns the unresolved correction rather than relearning the entire solution map. The intended distinctions are:

- The target Gibbs measure is defined by *FDM/FEM discrete energy*, not by an empirical data distribution alone.
- The deterministic limit $\beta \to \infty$ and the mesh–temperature coupling as $h \to 0$ are explicit.
- In linear SPD problems, residual whitening makes the target moment exactly $N(0, I)$.
- Mean–fluctuation factorization separates forcing from temperature effects and uses score distillation to test high-frequency physical calibration.
- The neural component models the remaining approximation error and acts as a learned stochastic corrector inside an existing numerical pipeline.

The random-map experiment in section 11 tests one narrow version of this practical claim: a stochastic correction model and a deterministic correction regressor use the same physical initializer, conditioning channels, small backbone, data splits, update budget, and validation-based checkpoint rule. The random maps are used to generate diverse physical configurations for the same solver, not to make geometry recognition itself the endpoint. Any procedure that produces a controlled family of coefficient fields, boundary conditions, forcing functions, geometries, initial states, observations, or material parameters could play the same conceptual role. The benchmark is adjacent to neural-operator transfer, learned discretization, solver-in-the-loop training, and residual-refinement diffusion [13, 12, 2, 21, 17]. The deterministic regressor is not a PINN and is not weak; it uses supervised high-fidelity correction targets and generalizes strongly. PINNs may instead use residual-based self-supervision, data terms, or Bayesian/ensemble extensions. The new result shows only that, in this low-data solver-error-law benchmark, diffusion correction gives a small but systematic improvement over this matched deterministic point estimator. It does not imply that deterministic regression, PINNs, neural operators, PDE-Refiner, or classical solvers cannot generalize under other designs or budgets.

3 Amortized Complexity Relative to Per-Instance PINN Optimization

The method should not be claimed to be universally faster or better than PINNs. The relative advantage depends on the problem class, training regime, available numerical infrastructure, whether a deterministic or probabilistic solution is required, and whether many related queries are solved after training. It is not that a diffusion model beats a direct sparse solver on a small linear system; it does not in the warm-start pilot. The possible structural advantage is different: a per-instance deterministic PINN usually solves each new PDE condition by online residual optimization, whereas SPADES performs offline amortized training and then uses a fixed number of denoising and correction steps for each new condition.

Let N_c be the number of collocation or quadrature points used by a PINN, let T_{opt} be the number of optimizer steps required for a target residual tolerance, and let C_{AD} denote the cost of one residual evaluation with the necessary automatic differentiation at the chosen network width, PDE order, and spatial dimension. A per-instance PINN solve has online cost

$$C_{\mathrm{PINN}}(c) = O(T_{\mathrm{opt}}\, N_c\, C_{\mathrm{AD}})\,. \tag{4}$$

When c changes, for example through a new coefficient field, forcing, boundary condition, or initializer, this optimization is repeated unless a separate operator-learning architecture is used.

After SPADES has been trained, one query has the online cost

$$C_{\mathrm{SPADES}}(c) = O(C_{\mathrm{init}} + K\, C_\theta(N) + M\, C_{\mathrm{corr}}(N))\,, \tag{5}$$

where C_{init} is the cost of the physical initializer, K is the number of reverse denoising steps, $C_\theta(N)$ is one denoiser forward pass on N degrees of freedom, M is the number of solver-native correction steps, and $C_{\mathrm{corr}}(N)$ is one such correction. On fixed-topology one-dimensional or structured-grid convolutional models, $C_\theta(N) = O(N)$ and Jacobi/Richardson-type corrections satisfy $C_{\mathrm{corr}}(N) = O(N)$. Thus, for fixed K, M and a cheap initializer, the online complexity is linear in the number of unknowns. In higher dimensions this remains architecture-dependent, but it is still a fixed-step inference cost rather than an online residual optimization loop.

Proposition 3.1 (Amortized speed condition)**.** *Consider Q independent PDE queries from the same condition family. Let C_{train} be the offline training cost of SPADES and write*

$$C_{\mathrm{online}}(N) = C_{\mathrm{init}} + K\, C_\theta(N) + M\, C_{\mathrm{corr}}(N).$$

The average SPADES cost per query is

$$\overline{C}_{\mathrm{SPADES}}(Q, N) = \frac{C_{\mathrm{train}}}{Q} + O(C_{\mathrm{online}}(N)). \tag{6}$$

If, along the target problem sequence,

$$C_{\mathrm{online}}(N) = o(T_{\mathrm{opt}}(N)\, N_c(N)\, C_{\mathrm{AD}}(N)) \quad \textit{and} \quad \frac{C_{\mathrm{train}}(N)}{Q} = o(T_{\mathrm{opt}}(N)\, N_c(N)\, C_{\mathrm{AD}}(N))\,, \tag{7}$$

then

$$\overline{C}_{\mathrm{SPADES}}(Q, N) = o(C_{\mathrm{PINN}}(N))\,.$$

Proof. For Q queries, a per-instance PINN repeats the optimization and therefore costs

$$C_{\mathrm{PINN,total}}(Q, N) = O(Q\, T_{\mathrm{opt}}(N)\, N_c(N)\, C_{\mathrm{AD}}(N))\,.$$

SPADES pays the training cost once and then performs fixed-step online inference:

$$C_{\mathrm{SPADES,total}}(Q, N) = C_{\mathrm{train}}(N) + O(Q\, C_{\mathrm{online}}(N)).$$

Dividing by Q gives equation (6). The two assumptions in equation (7) imply that both terms in equation (6)

are $o(T_{\text{opt}} N_c C_{\text{AD}})$. Combining this with equation (4) gives the claim. □

This proposition is a complexity statement, not an empirical benchmark. It says only that SPADES can have an amortized online advantage over per-instance PINN optimization when many related queries are solved, the trained denoiser generalizes, the initializer and corrector remain cheap, and the fixed-step inference cost satisfies the stated small-o condition. It does not compare against Bayesian PINNs, ensemble PINNs, operator-learning PINNs, DeepONets, FNOs, or classical sparse solvers, all of which require separate equal-accuracy and equal-wall-clock benchmarks.

4 Discrete Gibbs Formulation

4.1 General residual-based definition

After eliminating or strongly enforcing boundary conditions, let the discrete state be $z \in \mathbb{R}^{n_h}$. The condition c may include the domain, coefficient field, forcing, initial and boundary conditions, observations, and other inputs. Let the discrete residual be

$$R_h : \mathbb{R}^{n_h} \times \mathcal{C} \to \mathbb{R}^{m_h},$$

and let $W_h(c) \in \mathbb{R}^{m_h \times m_h}$ be a symmetric positive-definite residual metric.

Definition 4.1 (Discrete variational Gibbs measure)**.** Define the energy and Gibbs measure by

$$E_h(z;c) = \frac{1}{2} R_h(z;c)^{\mathsf{T}} W_h(c) R_h(z;c) + \Phi_h(z;c), \tag{8}$$

$$\pi_{\beta,h}(\mathrm{d}z \mid c) = \frac{1}{Z_{\beta,h}(c)} \exp[-\beta E_h(z;c)]\, \mathrm{d}z, \qquad Z_{\beta,h}(c) = \int_{\mathbb{R}^{n_h}} e^{-\beta E_h(z;c)} \mathrm{d}z. \tag{9}$$

Here $\beta > 0$ is the inverse temperature.

W_h determines units and mesh scaling. The choice $W_h = I$ gives a plain least-squares residual, whereas K^{-1} or a mass-matrix-based metric can be more natural for FEM. The term Φ_h may encode inequality constraints, regularization, constitutive priors, observation potentials, and similar effects.

Theorem 4.2 (Normalizability and low-temperature concentration)**.** *Fix h, c. Suppose E_h is continuous and, for some $a > 0, p > 0, b \in \mathbb{R}$, $E_h(z;c) \geq a \|z\|^p - b$. Then $Z_{\beta,h}(c) < \infty$ for all $\beta > 0$. If the minimizer set $\mathcal{M}_h(c) = \arg\min_z E_h(z;c)$ is nonempty and compact, then for any open set $U \supset \mathcal{M}_h(c)$,*

$$\pi_{\beta,h}(U^c \mid c) \longrightarrow 0, \qquad \beta \to \infty.$$

If the minimizer is unique, $\mathcal{M}_h(c) = \{z_h^*(c)\}$, then $\pi_{\beta,h} \Rightarrow \delta_{z_h^*}$.

Proof. Coercivity makes $e^{-\beta E_h}$ decay fast enough for a finite normalizing constant. By continuity and compactness, there is a positive energy gap Δ between U^c and the minimizer set. Lower-bound the denominator on a small ball around a minimizer and upper-bound the integral over U^c by an $e^{-\beta\Delta}$ factor. The probability tends to zero. If the minimizer is unique, mass concentrates in every neighborhood, which gives weak convergence. □

Remark 4.3. If a nonlinear PDE has multiple discrete solutions, the low-temperature measure concentrates on the minimizer set rather than on a single solution. The limiting weight of each branch depends on the local Hessian, symmetries, energy differences, and related quantities. In such cases, the multimodal representation capacity of diffusion may be intrinsically more useful than a deterministic Newton iteration.

4.2 Exact distribution for linear residuals

Theorem 4.4 (Linear least-squares Gibbs law)**.** *Let $R(z) = Az - b$, $W \succ 0$, and $H = A^{\mathsf{T}} W A \succ 0$. If $\Phi = 0$, then*

$$\pi_\beta(z \mid b) = \mathcal{N}(m_\beta, C_\beta), \qquad m_\beta = H^{-1} A^{\mathsf{T}} W b, \qquad C_\beta = \beta^{-1} H^{-1}.$$

In particular, if A is square and invertible, then $m_\beta = A^{-1}b$.

Proof. Complete the square in $\frac{1}{2}(Az-b)^\mathsf{T}W(Az-b)$ with respect to z: $\frac{1}{2}(z-m)^\mathsf{T}H(z-m)+\text{const}$. □

If A_h is a stable linear difference matrix in FDM, theorem 4.4 applies directly. For symmetric coercive FEM, however, it is often more natural to use the original variational functional than a squared residual.

4.3 SPD FEM variational Gibbs law

For a Poisson-type problem, define the discrete energy

$$J_h(z;b) = \frac{1}{2}z^\mathsf{T}Kz - b^\mathsf{T}z, \qquad K = K^\mathsf{T} \succ 0. \tag{10}$$

The classical FEM solution is $z_h^* = K^{-1}b$.

Theorem 4.5 (FEM Gaussian law and residual whitening)**.** *The Gibbs law associated with equation (10) is*

$$Z_h \mid b, \beta \sim \mathcal{N}(K^{-1}b, \beta^{-1}K^{-1}). \tag{11}$$

If $K = LL^\mathsf{T}$ is a Cholesky factorization, then

$$Z_h = z_h^* + \beta^{-1/2}L^{-\mathsf{T}}\xi, \qquad \xi \sim \mathcal{N}(0, I), \tag{12}$$

$$y = \sqrt{\beta}L^{-1}(KZ_h - b) = L^\mathsf{T}\sqrt{\beta}(Z_h - z_h^*) \qquad \sim\mathcal{N}(0, I). \tag{13}$$

Moreover,

$$J_h(z;b) - J_h(z_h^*;b) = \frac{1}{2}(z-z_h^*)^\mathsf{T}K(z-z_h^*) = \frac{1}{2}(Kz-b)^\mathsf{T}K^{-1}(Kz-b). \tag{14}$$

Proof. Use $Kz_h^* = b$ to complete the square in J_h. Substituting equation (12) into $K = LL^\mathsf{T}$ immediately gives equation (13). □

This result shows that the residual metric $W = K^{-1}$ is the choice that exactly matches the variational energy. The plain residual $\|Kz-b\|_2^2$ would produce covariance K^{-2}, which differs from the FEM energy geometry.

Corollary 4.6 (Temperature separation and equipartition)**.** *If $v = \sqrt{\beta}(Z_h - z_h^*)$, then $v \sim \mathcal{N}(0, K^{-1})$ and is independent of b and β. Also,*

$$\mathbb{E}\,\|Z_h - z_h^*\|_2^2 = \beta^{-1}\,\mathrm{tr}(K^{-1}), \tag{15}$$

$$\mathbb{E}\,\|Z_h - z_h^*\|_K^2 = \mathbb{E}[(Z_h - z_h^*)^\mathsf{T}K(Z_h - z_h^*)] = \frac{n_h}{\beta}. \tag{16}$$

5 Mesh–Temperature Coupling and Continuum Limit

If finite-temperature fluctuations are kept fixed under mesh refinement, the total energy spread may grow with the number of degrees of freedom. Therefore $h \to 0$ and $\beta_h \to \infty$ must be considered together.

Theorem 5.1 (Preservation of deterministic FEM order)**.** *Assume that the exact solution u and the Galerkin solution u_h^* satisfy $\|u - u_h^*\|_a \le Ch^p$ in the energy norm. Let U_h be the random FEM function whose coefficient vector follows equation (11). Then*

$$\mathbb{E}\,\|u - U_h\|_a^2 = \|u - u_h^*\|_a^2 + \frac{n_h}{\beta_h}. \tag{17}$$

Thus, if $\beta_h \gtrsim n_h h^{-2p}$, the mean-square energy error remains $O(h^{2p})$. On a quasi-uniform d-dimensional mesh where $n_h \asymp h^{-d}$, a sufficient condition is

$$\beta_h = \Omega\Big(h^{-(d+2p)}\Big). \tag{18}$$

Proof. The fluctuation $U_h - u_h^*$ has mean zero, so the expected cross term vanishes. The remaining fluctuation energy is n_h/β_h by equation (16). □

Remark 5.2 (Temperature calibration from error estimators)**.** If a reliable a posteriori energy-error estimator η_h is available, one can set

$$\beta_h = \frac{n_h}{\eta_h^2} \tag{19}$$

so that the expected stochastic spread in energy matches the estimated discretization error. This is a calibration principle, not an automatic guarantee of frequentist coverage.

6 Convergence Guarantees

There are three distinct convergence statements behind SPADES. First, the target Gibbs law itself concentrates on deterministic discrete minimizers as $\beta \to \infty$; this was proved in theorem 4.2. Second, if the exact time-dependent score of the noised Gibbs law is used, the reverse denoising dynamics reproduce the Gibbs target. Third, for convex energies or SPD linear systems, the physics relaxation or solver-native corrector contracts toward the corresponding Gibbs law or deterministic solution. The theorems below make these statements explicit. They do not assert that a finite neural network trained for finitely many steps has zero score error; the numerical diagnostics measure the remaining model and sampler bias.

Theorem 6.1 (Consistency of exact reverse denoising)**.** *Let $p_0 = \pi_{\beta,h}$ be a smooth positive density on $\mathbb{R}^{n_h}$ with sufficient decay, and let p_t be the law of the forward noising SDE*

$$\mathrm{d}X_t = f_t(X_t)\mathrm{d}t + g_t \mathrm{d}B_t, \qquad 0 \le t \le T, \tag{20}$$

where $g_t > 0$ is scalar and f_t is smooth. Assume p_t is smooth and positive for $t \in [0,T]$. Define the reverse-time process Y_s, $0 \le s \le T$, initialized with $Y_0 \sim p_T$, by

$$\mathrm{d}Y_s = \left[-f_{T-s}(Y_s) + g_{T-s}^2 \nabla \log p_{T-s}(Y_s)\right] \mathrm{d}s + g_{T-s}\mathrm{d}\bar{B}_s. \tag{21}$$

Then $Y_s \sim p_{T-s}$ for every $s \in [0,T]$. In particular, $Y_T \sim p_0 = \pi_{\beta,h}$.

Proof. The forward density satisfies the Fokker–Planck equation

$$\partial_t p_t = -\nabla \cdot (f_t p_t) + \frac{1}{2} g_t^2 \Delta p_t. \tag{22}$$

Set $q_s = p_{T-s}$. Then

$$\partial_s q_s = -\partial_t p_t|_{t=T-s} = \nabla \cdot (f_{T-s} q_s) - \frac{1}{2} g_{T-s}^2 \Delta q_s.$$

The Fokker–Planck equation for equation (21) is

$$\begin{aligned}
\partial_s q_s &= -\nabla \cdot \left(\left[-f_{T-s} + g_{T-s}^2 \nabla \log q_s\right] q_s\right) + \frac{1}{2} g_{T-s}^2 \Delta q_s \\
&= \nabla \cdot (f_{T-s} q_s) - g_{T-s}^2 \nabla \cdot (q_s \nabla \log q_s) + \frac{1}{2} g_{T-s}^2 \Delta q_s \\
&= \nabla \cdot (f_{T-s} q_s) - \frac{1}{2} g_{T-s}^2 \Delta q_s,
\end{aligned}$$

because $q_s \nabla \log q_s = \nabla q_s$. Thus the density curve $q_s = p_{T-s}$ solves the reverse Fokker–Planck equation with initial density $q_0 = p_T$. Uniqueness of the Fokker–Planck solution under the smoothness assumptions gives $Y_s \sim q_s = p_{T-s}$. □

Corollary 6.2 (Vanishing terminal, score, and discretization error)**.** *Assume the reverse drift in equation (21) is globally Lipschitz on $[0,T]$, and let approximate samplers use initial laws $\widehat{p}_T^{(n)}$, learned scores $\widehat{s}_t^{(n)}$, and Euler–Maruyama step sizes Δt_n. If*

$$W_2(\widehat{p}_T^{(n)}, p_T) \to 0, \qquad \sup_{t,x} \left\| \widehat{s}_t^{(n)}(x) - \nabla \log p_t(x) \right\| \to 0, \qquad \Delta t_n \to 0,$$

then the generated terminal laws converge to $\pi_{\beta,h}$ in W_2.

Proof. Couple the exact reverse process and the approximate-score reverse process with the same Brownian path and coupled initial states. If $\varepsilon_n = \sup_{t,x} \left\| \widehat{s}_t^{(n)}(x) - \nabla \log p_t(x) \right\|$ and L is a common Lipschitz constant for the drift, Gronwall's inequality gives a pathwise bound

$$\mathbb{E} \left\| Y_T - \widehat{Y}_T^{(n)} \right\|^2 \le C_T \left(W_2^2(\widehat{p}_T^{(n)}, p_T) + \varepsilon_n^2 \right)$$

for a constant C_T depending on T, L and $\sup_t g_t$. Euler–Maruyama strong convergence for globally Lipschitz drifts adds an error $o(1)$ as $\Delta t_n \to 0$. Therefore the W_2 distance between the approximate terminal law and the exact terminal law tends to zero. By theorem 6.1, the exact terminal law is $\pi_{\beta,h}$. □

Remark 6.3 (Gaussian terminal prior)**.** For the VP noising used by DDPM, $X_T = \sqrt{\bar{\alpha}_T} X_0 + \sqrt{1-\bar{\alpha}_T}\epsilon$ with $\epsilon \sim N(0, I)$. If $\mathbb{E}\|X_0\|^2 < \infty$, then coupling X_T and ϵ gives

$$W_2(p_T, N(0, I)) \le \sqrt{\bar{\alpha}_T} \left(\mathbb{E}\|X_0\|^2 \right)^{1/2} + \left| 1 - \sqrt{1-\bar{\alpha}_T} \right| \sqrt{n_h}.$$

Thus the usual Gaussian terminal prior becomes exact in the limit $\bar{\alpha}_T \to 0$, up to score and time-discretization errors.

Theorem 6.4 (Exponential convergence of Gibbs Langevin relaxation)**.** *Assume $E_h(\cdot; c)$ is twice differentiable and m-strongly convex:*

$$\nabla^2 E_h(z; c) \succeq mI, \qquad m > 0.$$

Let Z_t solve

$$\mathrm{d}Z_t = -\beta \nabla E_h(Z_t; c)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

and let μ_t be its law. Then $\pi_{\beta,h}$ is invariant. Moreover, for any two initial laws μ_0, ν_0 with finite second moment,

$$W_2(\mu_t, \nu_t) \le e^{-\beta m t} W_2(\mu_0, \nu_0). \tag{23}$$

In particular, taking $\nu_0 = \pi_{\beta,h}$ gives

$$W_2(\mu_t, \pi_{\beta,h}) \le e^{-\beta m t} W_2(\mu_0, \pi_{\beta,h}).$$

Proof. Invariance follows from the generator

$$\mathcal{L}\varphi = -\beta \nabla E_h \cdot \nabla \varphi + \Delta \varphi.$$

For $\pi_{\beta,h}(\mathrm{d}z) \propto e^{-\beta E_h(z;c)}\mathrm{d}z$, integration by parts gives $\int \mathcal{L}\varphi \, \mathrm{d}\pi_{\beta,h} = 0$ for smooth compactly supported φ, hence $\pi_{\beta,h}$ is invariant.

For contraction, couple two solutions $Z_t, \widetilde{Z}_t$ with the same Brownian motion. Their difference $D_t = Z_t - \widetilde{Z}_t$ satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} \|D_t\|^2 = -2\beta \left\langle D_t, \nabla E_h(Z_t; c) - \nabla E_h(\widetilde{Z}_t; c) \right\rangle.$$

Strong convexity implies

$$\left\langle D_t, \nabla E_h(Z_t; c) - \nabla E_h(\widetilde{Z}_t; c) \right\rangle \ge m \|D_t\|^2.$$

Therefore $\|D_t\|^2 \le e^{-2\beta m t} \|D_0\|^2$. Taking expectations and then the infimum over all couplings of μ_0, ν_0 yields equation (23). □

Theorem 6.5 (Convergence of SPD solver-native correctors)**.** *Let $K = K^\mathsf{T} \succ 0$, $z^* = K^{-1}b$, and let $B = B^\mathsf{T} \succ 0$ be a preconditioner. Consider the correction*

$$z^{k+1} = z^k - \omega B^{-1}(Kz^k - b). \tag{24}$$

If

$$0 < \omega < \frac{2}{\lambda_{\max}(B^{-1/2} K B^{-1/2})},$$

then $z^k \to z^*$ linearly. More precisely, with $e^k = z^k - z^*$ and $\widetilde{e}^k = B^{1/2}e^k$,

$$\left\|\widetilde{e}^k\right\|_2 \le \rho^k \left\|\widetilde{e}^0\right\|_2, \qquad \rho = \max_i \left|1 - \omega\lambda_i(B^{-1/2}KB^{-1/2})\right| < 1. \tag{25}$$

Proof. Since $Kz^* = b$,

$$e^{k+1} = (I - \omega B^{-1}K)e^k.$$

Multiplying by $B^{1/2}$ gives

$$\widetilde{e}^{k+1} = \left(I - \omega B^{-1/2}KB^{-1/2}\right)\widetilde{e}^k.$$

The matrix $B^{-1/2}KB^{-1/2}$ is symmetric positive definite, so it has positive eigenvalues λ_i. The assumed range of ω makes every eigenvalue $1 - \omega\lambda_i$ of the iteration matrix lie strictly inside $(-1, 1)$. The spectral norm of the iteration matrix is therefore $\rho < 1$, giving equation (25) and linear convergence to z^*. □

7 Diffusion and Score Learning

7.1 Gibbs score and stochastic relaxation

The score of the general Gibbs law is

$$\nabla_z \log \pi_{\beta,h}(z \mid c) = -\beta\nabla_z E_h(z; c) = -\beta J_{R_h}(z; c)^{\mathsf{T}} W_h R_h(z; c) - \beta\nabla_z \Phi_h(z; c). \tag{26}$$

Consequently, the overdamped Langevin process

$$\mathrm{d}Z_s = -\beta\nabla E_h(Z_s; c)\mathrm{d}s + \sqrt{2}\mathrm{d}B_s \tag{27}$$

has $\pi_{\beta,h}$ as its stationary distribution. Thus denoising is not merely visual noise removal; it is stochastic relaxation toward a low-discrete-physics-energy manifold.

Langevin dynamics could be run separately for each condition, but in repeated problems a score or reverse transition can be amortized by a neural network. This work uses the variance-preserving DDPM forward process

$$q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I\big) \tag{28}$$

and the noise-prediction objective

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{x_0,t,\epsilon} \left\|\epsilon - \epsilon_\theta(x_t, t, c)\right\|_2^2, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{29}$$

from [8, 20].

7.2 Physical-initializer multi-condition residual diffusion

In practical FDM/FEM computation, a useful physical state is often available before denoising from pure white noise. Examples include a coarse-grid solution, a state from a previous physical time, a linearized solution, a harmonic lift of the boundary values, a sensor interpolation field, or a vector after a few classical iterations. Let this initializer be

$$z_h^{\mathrm{init}} = G_{0,h}(c). \tag{30}$$

The practical variant in this draft uses as the diffusion target not the full field but the normalized correction

$$r_0 = \frac{z_h^\star - z_h^{\mathrm{init}}}{s_r}, \qquad \delta z_h = z_h^\star - z_h^{\mathrm{init}}, \qquad s_r > 0. \tag{31}$$

Equivalently, the physical field is represented as

$$z_h = z_h^{\mathrm{init}} + \delta z_h.$$

The strongest practical claim of SPADES is this reference-centered decomposition. The neural model does not replace the numerical solver; it models the remaining approximation error and acts as a learned stochastic corrector inside an existing numerical pipeline. A typical pipeline is

$$\begin{aligned}&\text{FDM/FEM initializer} \to \text{residual extraction} \to \text{conditional diffusion}\\ &\to \text{Jacobi/CG/multigrid correction} \to \text{boundary or constraint projection.}\end{aligned}$$

The initializer can be a coarse-grid FDM solution, a coarse FEM solution, a multigrid prolongation, a reduced-order approximation, a low-fidelity simulation, a previous time-step solution, or another physics-based prior. When the initializer is informative, the learned target may have smaller dynamic range than the full solution field; this is an architectural and statistical advantage to be tested, not a universal guarantee of lower training cost.

The condition tensor can be formed as

$$\gamma_h = \left[z_h^{\text{init}},\ R_h(z_h^{\text{init}}; c),\ a_h,\ f_h,\ g_h,\ x_h,\ m_h, \mathcal{O}_h\right]. \tag{32}$$

Here a_h is a material property or PDE coefficient, f_h is a source, g_h denotes boundary values, x_h gives coordinates, m_h is a boundary or observation mask, and $\mathcal{O}_h$ denotes optional observations. Thus "multiple inputs" means combining physically different fields and scalars on a common grid or mesh graph, not concatenating a few unrelated scalar values.

The forward process is defined in correction coordinates by

$$q(r_t \mid r_0) = \mathcal{N}\big(\sqrt{\bar\alpha_t} r_0, (1-\bar\alpha_t) I\big)\,, \qquad z_t = P_B\big(z_h^{\text{init}} + s_r r_t\big)\,. \tag{33}$$

Here P_B is a projection that exactly enforces Dirichlet conditions and hard constraints. Even at large diffusion times, the physical state is "initializer plus normalized perturbation," so training occurs in a narrower dynamic range than full-field white noise.

This structure also preserves mature numerical infrastructure. Sparse matrix assembly, conservation-form residuals, preconditioners, multigrid hierarchies, boundary enforcement, finite-element projections, and adaptive-mesh indicators can remain part of the solver. The learned component supplies a stochastic correction proposal and optional distributional information; the discrete operator and constraint projection remain responsible for hard physical consistency.

Inference need not start at the terminal noise level T. If the correction of the initializer is set to zero and the chain starts at an intermediate noise level t_s,

$$\widetilde{r}_{t_s} = \sqrt{1-\bar\alpha_{t_s}}\,\epsilon, \qquad \widetilde{z}_{t_s} = P_B\big(z_h^{\text{init}} + s_r \widetilde{r}_{t_s}\big)\,, \tag{34}$$

then denoising begins after perturbing the neighborhood of the physical initializer rather than the full field. This initialization is generally not the exact DDPM marginal; it is a warm-start heuristic trading computation for prior mismatch.

Proposition 7.1 (Truncated-start mismatch)**.** *For a fixed target correction r_0, the forward marginal and the initial law in equation (34) are*

$$q_{t_s} = \mathcal{N}\big(\sqrt{\bar\alpha_{t_s}} r_0, (1-\bar\alpha_{t_s}) I\big)\,, \qquad \widetilde{q}_{t_s} = \mathcal{N}(0, (1-\bar\alpha_{t_s}) I)\,.$$

Therefore,

$$W_2(q_{t_s}, \widetilde{q}_{t_s}) = \sqrt{\bar\alpha_{t_s}}\,\|r_0\|_2\,, \qquad \mathrm{KL}(q_{t_s}\|\widetilde{q}_{t_s}) = \frac{\bar\alpha_{t_s}}{2(1-\bar\alpha_{t_s})}\,\|r_0\|_2^2\,. \tag{35}$$

The mismatch is small when the initializer is accurate, so $\|r_0\|$ is small, or when the chain starts at a sufficiently high noise level. Conversely, a poor coarse solution or too small a start level may create bias.

Proof. The two Gaussians have the same covariance. The W_2 distance is therefore the Euclidean norm of the mean difference, and the KL divergence is the squared mean difference measured by the inverse common covariance. □

7.3 Condition-wise reverse-start selection from the initial residual

A fixed t_s assumes that the physical initializer has the same quality for every condition. In practice, coefficient contrast, forcing frequency, and boundary layers can make the error of z_h^{init} vary substantially. For the linear SPD system

$$A_h z_h = b_h, \qquad A_h = A_h^{\mathsf{T}} \succ 0, \tag{36}$$

define the initial residual and exact correction by

$$q_h = A_h z_h^{\text{init}} - b_h, \qquad e_h = z_h^\star - z_h^{\text{init}} = -A_h^{-1} q_h. \tag{37}$$

Then the identity

$$\|e_h\|_{A_h}^2 = q_h^{\mathsf{T}} A_h^{-1} q_h \tag{38}$$

holds exactly. To avoid re-solving the fine problem, we approximate the correction with a cheap preconditioner or with J weighted-Jacobi sweeps:

$$\widehat{e}_h = -M_h^{-1} q_h, \qquad \widehat{Q}_h(c) = \frac{\|\widehat{e}_h\|_2}{s_r \sqrt{n_I}}, \tag{39}$$

where n_I is the number of interior degrees of freedom. Let the diffusion schedule's noise-to-signal ratio be

$$\rho_t = \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}}. \tag{40}$$

For a safety factor $\kappa \geq 1$ and an admissible set of reverse-start steps $\mathcal{T}$, choose

$$t_s(c) = \arg\min_{t \in \mathcal{T}} \left| \log \rho_t - \log\Big(\kappa \widehat{Q}_h(c) + \varepsilon\Big) \right|. \tag{41}$$

The choice $\kappa = 1$ approximately matches the estimated correction RMS and the diffusion noise RMS. Larger κ starts from a noisier level and is more conservative against underestimating the initializer error.

Proposition 7.2 (Monotonicity and mismatch control of adaptive starts)**.** *Assume $\bar{\alpha}_t$ decreases with t and $\mathcal{T}$ is a consecutive set of schedule steps. Then the nearest-level rule equation (41) is nondecreasing in $\widehat{Q}_h$: a condition with a larger residual-error proxy starts at a larger diffusion index. Let the true normalized correction RMS be*

$$Q_h = \frac{\|e_h\|_2}{s_r \sqrt{n_I}}.$$

If $Q_h \leq c\widehat{Q}_h$ and the schedule grid contains the exact level $\rho_{t_s} = \kappa \widehat{Q}_h$, then the truncated-start Gaussian mismatch per interior degree of freedom satisfies

$$\frac{1}{n_I} \mathrm{KL}(q_{t_s} \| \widetilde{q}_{t_s}) = \frac{Q_h^2}{2\rho_{t_s}^2} \leq \frac{c^2}{2\kappa^2}. \tag{42}$$

Proof. Because ρ_t increases as $\bar{\alpha}_t$ decreases, the nearest projection onto the ordered grid $\{\log \rho_t : t \in \mathcal{T}\}$ is nondecreasing in the input $\log(\kappa \widehat{Q}_h + \varepsilon)$. Thus a larger proxy cannot select an earlier, less noisy start level.

For the mismatch bound, equation (35) gives

$$\mathrm{KL}(q_{t_s} \| \widetilde{q}_{t_s}) = \frac{\bar{\alpha}_{t_s}}{2(1-\bar{\alpha}_{t_s})} \|r_0\|_2^2.$$

Here $\|r_0\|_2^2 = n_I Q_h^2$ and $\bar{\alpha}_t/(1-\bar{\alpha}_t) = \rho_t^{-2}$, so

$$\frac{1}{n_I} \mathrm{KL}(q_{t_s} \| \widetilde{q}_{t_s}) = \frac{Q_h^2}{2\rho_{t_s}^2}.$$

Substituting $Q_h \leq c\widehat{Q}_h$ and $\rho_{t_s} = \kappa \widehat{Q}_h$ proves the bound. □

Remark 7.3. The adaptive rule in equation (41) is a cheap deterministic warm-start policy. The exact

linear estimator $M_h^{-1} = A_h^{-1}$ is useful for diagnostics but removes the computational benefit. For finite-temperature Gibbs targets, the desired thermal fluctuation is present in addition to the deterministic error e_h; directly combining this rule with a residual-minimizing corrector can shrink the intended covariance. The v0.5 implementation therefore treats residual-adaptive starts primarily as a deterministic-target path.

Proposition 7.4 (Score transformation in residual coordinates)**.** *Let $z = z_h^{\mathrm{init}} + s_r r$, and suppose $p_z(z \mid c)$ has a density. The induced density satisfies*

$$p_r(r \mid c) = s_r^{n_h} p_z(z_h^{\mathrm{init}} + s_r r \mid c), \qquad \nabla_r \log p_r(r \mid c) = s_r \nabla_z \log p_z(z \mid c). \tag{43}$$

Thus the warm-start coordinate change preserves the target distribution while changing the center and scale of the score.

Proof. Apply the affine change-of-variables formula and the chain rule. □

Proposition 7.5 (Reference invariance of correction coordinates)**.** *Let $T_c(r) = z_h^{\mathrm{init}}(c) + s_r r$ with $s_r > 0$, and let $q_r(\mathrm{d}r \mid c)$ be the pullback of a physical target law $\pi_z(\mathrm{d}z \mid c)$ by T_c. Then $T_{c\#} q_r = \pi_z$. Changing the initializer changes the coordinate distribution over corrections, but not the represented physical law, provided the affine map is used exactly on the free degrees of freedom.*

Proof. The map T_c is an invertible affine transformation on the free discrete coordinates. The statement follows directly from the definition of pushforward and pullback measures; the constant Jacobian is absorbed into the normalizing density. □

Remark 7.6. Finite neural capacity and finite training data are not invariant to this coordinate change. A poor initializer can make δz_h large and difficult to learn, while an overly accurate high-fidelity initializer can make the learned correction unnecessary. The initializer must therefore be reported as part of the method.

The predicted clean correction is obtained from the noise predictor by

$$\widehat{r}_0(r_t, t, \gamma_h) = \frac{r_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(r_t, t, \gamma_h)}{\sqrt{\bar{\alpha}_t}}. \tag{44}$$

When training an amortized solver for a deterministic PDE family, the implementation uses

$$\begin{aligned}\mathcal{L}_{\mathrm{warm}} = \mathcal{L}_{\mathrm{DSM}} &+ \lambda_0 \mathbb{E}\Big[w_t \left\|\widehat{r}_0 - r_0\right\|_2^2\Big] \\ &+ \lambda_P \mathbb{E}\left[\mathbf{1}_{t \le t_P} \frac{\left\|D_h^{-1} R_h(P_B(z_h^{\mathrm{init}} + s_r \widehat{r}_0); c)\right\|_2^2}{s_r^2}\right].\end{aligned} \tag{45}$$

Here D_h is the diagonal of A_h or another low-cost preconditioner. The physics term is applied only to the clean-correction estimate, not directly to noisy r_t, and it is restricted to late noise levels to avoid numerical blow-up at large t.

If the target is a finite-temperature Gibbs sample, reusing the residual-zero term in equation (45) can collapse the variance to the deterministic solution. In that case, one should use a residual-distribution loss such as equation (51), exact or approximate Gibbs samples, or post-hoc physics distillation. The warm-start parameterization can serve both the deterministic limit and the finite-temperature law, but the physics loss must differ between them.

At a reverse step, the predicted clean correction is adjusted once by a classical iteration. For a linear system $A_h z = b_h$,

$$\begin{aligned}\widetilde{z}_0 &= P_B(z_h^{\mathrm{init}} + s_r \widehat{r}_0), \\ z_0^+ &= P_B\big(\widetilde{z}_0 - \omega D_h^{-1}(A_h \widetilde{z}_0 - b_h)\big), \qquad r_0^+ = \frac{z_0^+ - z_h^{\mathrm{init}}}{s_r}\end{aligned} \tag{46}$$

is used, and r_0^+ is then inserted into the next DDIM/DDPM mean calculation. This is a predictor–corrector structure: diffusion proposes a global or nonlinear correction, and FDM/FEM relaxation reduces local

physical inconsistency. Jacobi can be replaced by Gauss–Seidel, a multigrid V-cycle, projection, or a Newton–Krylov step.

7.4 Why direct coefficient diffusion is unstable

If a conditional DDPM is trained on normalized z_h directly, one network must simultaneously represent

1. the large forcing-dependent mean shift $K^{-1}b$,
2. the low-frequency-dominated covariance K^{-1},
3. the small variance change $\beta^{-1/2}$,
4. and the component-scale differences near mesh boundaries.

Even when the mean error looks small, high-frequency errors amplified by K can make the whitened residual badly wrong. During development, the direct coefficient model diverged in reverse sampling, and finite-temperature calibration remained poor even after clipping. This negative result motivates the factorized formulation.

7.5 Mean–fluctuation factorization

By theorem 4.6, a linear SPD family decomposes exactly as

$$z = z_h^*(b) + \beta^{-1/2} v, \qquad v \sim \mathcal{N}(0, K^{-1}). \tag{47}$$

The experiments train two modules.

Mean operator. Let g_ϕ predict the normalized mean coefficient $\widetilde{z}_h^*$ from the normalized load $\widetilde{b}$. We minimize

$$\mathcal{L}_{\text{mean}} = \mathbb{E}\left\| g_\phi(\widetilde{b}) - \widetilde{z}_h^* \right\|_2^2 + \lambda_m \mathbb{E}\left\| L^{-1}(K\widehat{z}_h^* - b) \right\|_2^2 . \tag{48}$$

The first term is operator supervision and the second is a Galerkin residual. The first term can be removed in a fully unsupervised setting, but the identifiability and optimization difficulty change.

Temperature-independent fluctuation diffusion. The training target is $v = \sqrt{\beta}(z - z_h^*)$. We normalize $x_0 = D^{-1}v$ by component standard deviations D and train a DDPM. At inference,

$$\widehat{z} = g_\phi(b) + \beta^{-1/2} D \widehat{x}_0 \tag{49}$$

recovers the field. This structure enforces temperature scaling exactly instead of asking the network to memorize it.

7.6 Whitened-residual moment loss

For a predicted denoised fluctuation $\widehat{v}_0$, compute

$$\widehat{y} = L^{\mathsf{T}} \widehat{v}_0. \tag{50}$$

Since the target is $\mathcal{N}(0, I)$, use the minibatch mean $\overline{y}$ and covariance $\widehat{C}_y$ to define

$$\mathcal{L}_{\text{mom}} = \|\overline{y}\|_2^2 + \frac{1}{n_h^2} \left\| \widehat{C}_y - I \right\|_F^2 . \tag{51}$$

In the implementation, this loss is applied only at low diffusion times, and clipping or robust transforms can keep early outliers from overwhelming DSM.

The key point is that this is a *residual distribution loss*, not a residual-zero loss. Minimizing only $\|Kz - b\|^2$ collapses to $z = z_h^*$, whereas equation (51) preserves the variance required at finite temperature.

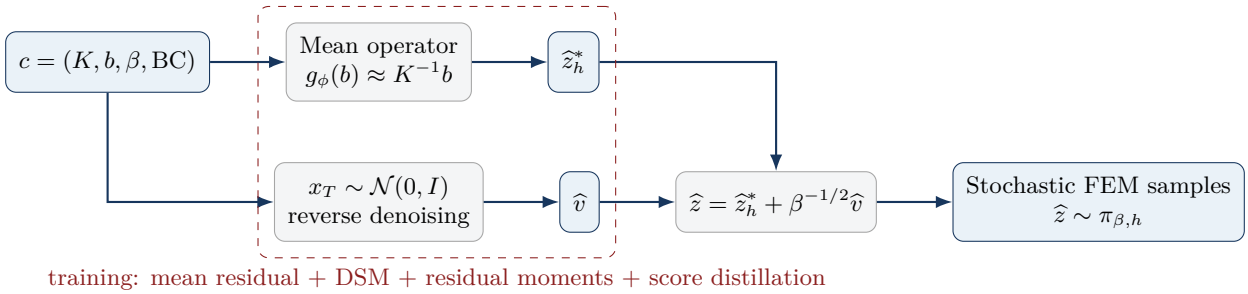


Figure 1: Training and inference flow of factorized SPADES. For linear SPD problems, forcing and temperature can be separated. In general nonlinear problems, the fluctuation model depends on the condition c and may need to represent a multimodal distribution.

7.7 Linear Gaussian score distillation

For the Poisson verification problem, the fluctuation law is Gaussian, so the optimal noised-marginal score can be computed in closed form.

Proposition 7.7 (Optimal Gaussian noise predictor)**.** *If $x_0 \sim \mathcal{N}(0, C)$ and equation (28) is applied, then*

$$\Sigma_t = \bar{\alpha}_t C + (1 - \bar{\alpha}_t) I. \tag{52}$$

The MSE-optimal noise predictor is

$$\epsilon^*(x_t, t) = W_t^* x_t, \qquad W_t^* = \sqrt{1 - \bar{\alpha}_t}\, \Sigma_t^{-1}, \tag{53}$$

and the marginal score is $s_t^*(x_t) = -\Sigma_t^{-1} x_t$.

Proof. The pair (ϵ, x_t) is jointly Gaussian and $\mathrm{Cov}(\epsilon, x_t) = \sqrt{1 - \bar{\alpha}_t} I$, so the Gaussian conditional mean formula applies. □

For a time-indexed linear denoiser $W_{\theta,t}$, add

$$\mathcal{L}_{\mathrm{SD}} = \frac{1}{T} \sum_{t=1}^{T} \| W_{\theta,t} - W_t^* \|_F^2 + \frac{1}{T} \sum_t \| b_{\theta,t} \|_2^2 . \tag{54}$$

The toy experiment initializes from equation (53) and then fine-tunes with DSM. This is an oracle unavailable in general nonlinear problems, so the result should be interpreted as a theorem check of whether the theory and sampler agree. In nonlinear problems, the oracle can be replaced by the energy gradient in equation (26), solver samples, or score-matching identities.

The final fluctuation loss is

$$\mathcal{L}_{\mathrm{fluc}} = \mathcal{L}_{\mathrm{DSM}} + \lambda_r \mathcal{L}_{\mathrm{mom}} + \lambda_s \mathcal{L}_{\mathrm{SD}}. \tag{55}$$

8 Algorithms

Algorithm 1: Linear SPD SPADES training

1. Factorize $K = LL^{\mathsf{T}}$ and generate conditions b_i and temperatures β_i.
2. Compute $z_i^* = K^{-1}b_i$ with a high-fidelity or exact solver.
3. Draw $\xi_i \sim N(0, I)$ and generate $z_i = z_i^* + \beta_i^{-1/2} L^{-\mathsf{T}}\xi_i$.
4. Train the mean operator g_ϕ with equation (48).
5. Construct $v_i = \sqrt{\beta_i}(z_i - z_i^*)$ and the normalized target $x_{0,i} = D^{-1}v_i$.
6. For random t, ϵ, generate $x_t = \sqrt{\bar\alpha_t}x_0 + \sqrt{1-\bar\alpha_t}\epsilon$.
7. Compute $\widehat{x}_0 = (x_t - \sqrt{1-\bar\alpha_t}\epsilon_\theta)/\sqrt{\bar\alpha_t}$ and minimize equation (55).
8. At inference, sample $\widehat{v}$ by reverse diffusion and assemble with equation (49).

Algorithm 2: Multi-input warm-start solver with a physical initializer

1. Compute $z_h^{\mathrm{init}} = G_{0,h}(c)$ using coarse FDM/FEM, a previous physical state, or a boundary lift.
2. Build $\gamma_h = [z_h^{\mathrm{init}}, R_h(z_h^{\mathrm{init}}), a_h, f_h, g_h, x_h, m_h, \mathcal{O}_h]$ and set the correction scale s_r from the training set.
3. Add noise to the target correction $r_0 = (z_h^\star - z_h^{\mathrm{init}})/s_r$ and train the conditional denoiser with equation (45).
4. At inference, start from $r_T \sim N(0, I)$ for the full horizon. To exploit an accurate initializer, start at $t_s < T$ using equation (34). The start level can be a fixed validation-chosen value, or it can be selected condition-wise from the residual proxy in equation (41). Always interpret the physical state as $z_t = P_B(z_h^{\mathrm{init}} + s_r r_t)$.
5. In late reverse steps, apply equation (46) one or more times and use the corrected r_0^+ to compute the next DDIM/DDPM step.
6. Output the mean and variance over multiple initial noises. If only deterministic training targets were used, do not interpret this variance as a calibrated physical posterior.

Algorithm 3: General nonlinear residual-Gibbs learning and sampling

1. Define $E_h(z; c) = \frac{1}{2}R_h^{\mathsf{T}}W_hR_h + \Phi_h$ and implement the differentiable gradient $\nabla_z E_h$.
2. Construct target samples from solver ensembles, continuation, Langevin/MALA, or observational data. If no data are available, use energy-based score matching or annealed Langevin.
3. Train the conditional score $s_\theta(z_t, t, c)$ with DSM, adding $-\beta\nabla E_h$ consistency and boundary or conservation-law projections.
4. After each reverse SDE/ODE step, apply a physics corrector $z \leftarrow z - \eta\beta\nabla E_h(z; c) + \sqrt{2\eta}\,\xi$. Use MALA accept/reject if the exact invariant law is required.
5. Sample from several initial noises to estimate multiple branches and posterior uncertainty.

9 One-Dimensional Poisson FEM Verification

9.1 Problem and data

We use

$$-u''(x) = f(x), \quad x \in (0,1), \qquad u(0) = u(1) = 0. \tag{56}$$

The discretization uses a uniform mesh with 32 interior degrees of freedom and P1 basis functions. The stiffness and mass matrices are

$$K = \frac{1}{h}\,\mathrm{tridiag}(-1, 2, -1), \qquad M = \frac{h}{6}\,\mathrm{tridiag}(1, 4, 1), \qquad h = \frac{1}{33}.$$

Table 2: Main distributional metrics. DSM-only uses the same factorization but trains without analytic score initialization or distillation. Oracle denotes the exact Gaussian sampler. Metrics marked ↓ are better when smaller; metrics marked → 1 are better when closer to 1.

Metric	Factorized DSM-only	SPADES score-distilled	Analytic Gibbs oracle
Score matrix relative error ↓	0.3198	**0.0071**	0
Whitened covariance error ↓	1.8166	**0.1374**	0.0156
Equipartition ratio → 1	2.2924	**1.0367**	1.0006
Conditional coefficient covariance error ↓	0.0526	0.0554	**0.0350**
Temperature slope → −0.5	-0.5000	**-0.5000**	-0.5000

The forcing is generated as

$$f(x) = \sum_{m=1}^{8} a_m m^{-1/2} \sin(m\pi x), \qquad a_m \overset{\text{iid}}{\sim} \mathcal{N}(0,1), \tag{57}$$

and $b = Mf_h$ is used. Of 8,512 total conditions, 8,000 are used for training and 512 for testing. $\log \beta$ is sampled uniformly from $[\log 10^4, \log 10^6]$.

9.2 Model and training setup

The mean operator is an affine layer mapping a normalized 32-dimensional load to a 32-dimensional mean coefficient. Because the Poisson map is linear, a larger architecture is unnecessary. Training uses AdamW with initial learning rate 2×10^{-3}, cosine decay, 750 steps, batch size 512, and $\lambda_m = 0.1$.

The fluctuation denoiser has a 32×32 linear matrix and bias for each of 64 diffusion times. Since the Bayes-optimal denoiser for a Gaussian target is linear, this architecture has no representation error. Training uses a cosine noise schedule, AdamW with initial learning rate 10^{-3}, 1,000 steps, batch size 512, $\lambda_r = 0.01$, and $\lambda_s = 100$. The model is initialized by W_t^* and fine-tuned with DSM. The reverse variance is the DDPM fixed-large choice, i.e. the forward β_t. The posterior lower-bound variance showed under-dispersion in this Gaussian test.

All experiments were run in a CPU-only container with PyTorch 2.10.0, Python 3.13.5, and at most 8 threads. Recorded wall-clock times were approximately 6.42 seconds for mean training and 11.72 seconds for fluctuation training. The hardware model was not identifiable inside the container, so absolute times should be treated as reference values.

9.3 Evaluation metrics

We use 64 test conditions and 2,048 shared fluctuation samples. The metrics are:

- Mean-operator relative error: $\mathbb{E}_i \|\widehat{z}_i^* - z_i^*\| / \|z_i^*\|$.
- Whitened-residual mean RMS and covariance relative Frobenius error: $\left\|\widehat{C}_y - I\right\|_F / \|I\|_F$.
- Equipartition ratio: $\mathbb{E}\|y\|^2 / n_h$, whose target is 1.
- Coefficient covariance error: $\left\|\widehat{C}_z - \beta^{-1}K^{-1}\right\|_F / \left\|\beta^{-1}K^{-1}\right\|_F$.
- Score-matrix error: $\|W_\theta - W^*\|_F / \|W^*\|_F$.
- Temperature slope: the slope from regressing log RMS fluctuation on $\log \beta$; the theoretical value is $-1/2$.

9.4 Results

In table 2, score distillation substantially improves high-frequency residual geometry. The DSM-only coefficient covariance error looks small at 0.0526, but its whitened covariance error is 1.8166 and its energy is 2.29 times the target. The reason is that the Frobenius norm of K^{-1} is dominated by low-frequency modes, so coefficient-space metrics hide high-frequency error. Residual whitening is an essential diagnostic.

Table 3: Mean and timing metrics for the score-distilled model. The sample-mean error includes finite-sample Monte Carlo error from 2,048 samples.

Metric	Value
Mean operator relative L^2 error	1.281×10^{-5}
Mean operator residual RMS	5.957×10^{-6}
Generated sample-mean relative L^2 error	1.031×10^{-3}
Generated sample-mean RMSE	3.295×10^{-5}
64 conditions $\times$ 2,048 shared samples, SPADES	0.563 s
Same analytic Gaussian sampling	0.416 s

The score-distilled model has energy ratio 1.0367, corresponding to 3.7% over-dispersion, and whitened covariance error 0.1374. Since this is much larger than the oracle finite-sample error of 0.0156, sampler bias clearly remains. The causes are the 64-step discrete reverse process, fixed isotropic reverse variance, and small residual score error. The result is not only a success case; it also quantifies where the method is not yet exact Gibbs sampling.

9.5 Interpretation

Deterministic limit. The temperature slope -0.500 is not a network-fitting result; it follows structurally from factorization. Thus the scaling itself persists outside the training range of β. Mean-operator error and fluctuation-shape error remain separate issues.

Meaning of probability. This Gaussian spread is not a posterior inferred from observational noise or physical parameter uncertainty. It is the numerical law defined by the chosen Gibbs energy and β. It should not be called a 90% confidence interval for the true solution. Actual uncertainty quantification requires a hierarchical combination of a distribution over c, an observation likelihood, and model discrepancy.

Computational cost. For the small linear Poisson problem, Cholesky sampling is simpler and slightly faster. SPADES is an unnecessary approximation in this case. A learning-based approach is justified only when at least one of the following is present: many repeated queries, nonlinear or non-Gaussian targets, partial-observation conditioning, multiple admissible branches under the chosen formulation, or complex model discrepancy.

10 Variable-Coefficient Poisson Warm-Start Pilot

10.1 Problem, inputs, and initializer

As a minimal validation of the practical variant, we solve

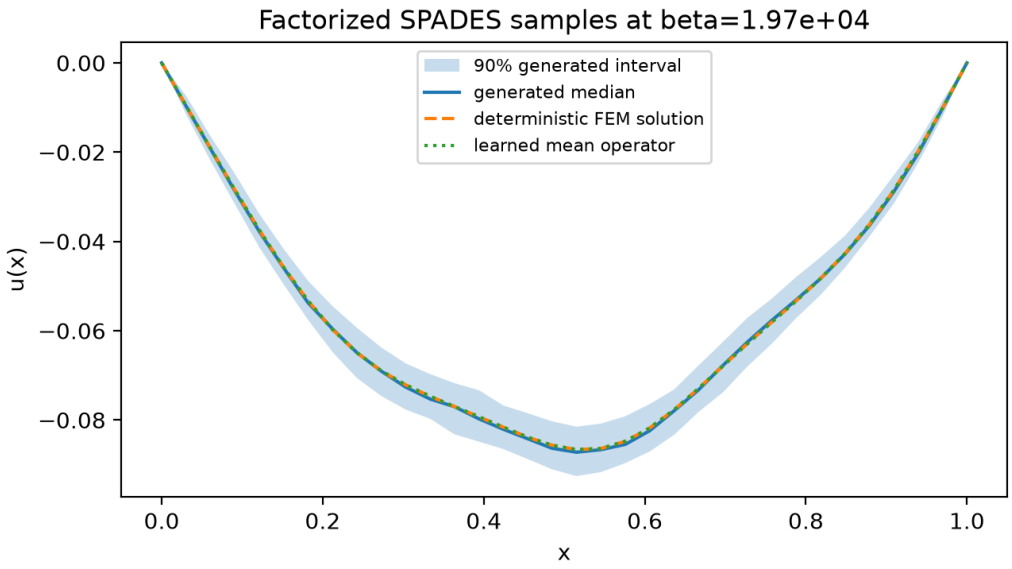
$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(k(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right) = f(x), \qquad x \in (0,1), \qquad u(0) = g_0, \quad u(1) = g_1 \tag{58}$$

with a 64-node conservative FDM discretization. The fields $\log k$ and f are generated from four decaying Fourier modes each, and $\log k$ is clipped to $[-1, 1]$. Boundary values are sampled independently from $[-0.6, 0.6]$. A fine tridiagonal solve is used as the training target, while the same input is solved on a 10-node coarse grid and linearly interpolated to form u^{init}. The train/validation/test split is 2048/256/256, and the final table uses a fixed set of 128 test cases.

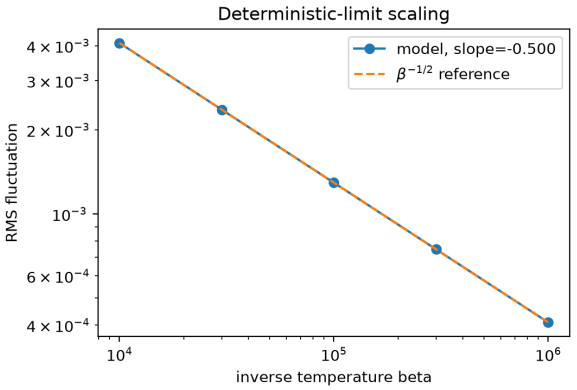
The eight channels of the condition tensor are

$$\gamma = \left[u^{\mathrm{init}},\ k,\ f,\ R_h(u^{\mathrm{init}}),\ x,\ g_0\mathbf{1},\ g_1\mathbf{1},\ m_{\mathrm{int}}\right].$$

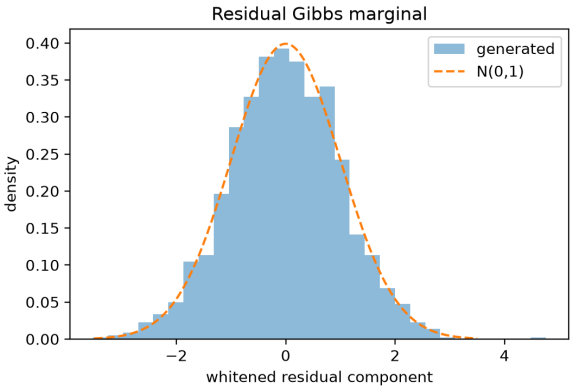
The target is $r_0 = (u^\star - u^{\mathrm{init}})/s_r$, with training-set RMS $s_r = 6.918 \times 10^{-3}$. This directly implements denoising from an initial physical state using several physical input fields.



(a) Generated 90% interval, median, deterministic FEM mean, and learned mean operator for one test forcing.



(b) Exact $\beta^{-1/2}$ scaling induced by temperature separation.



(c) Marginal distribution of representative whitened-residual components.

Figure 2: SPADES Poisson verification results. The mean overlaps the deterministic FEM solution, while the fluctuation width decreases with temperature.

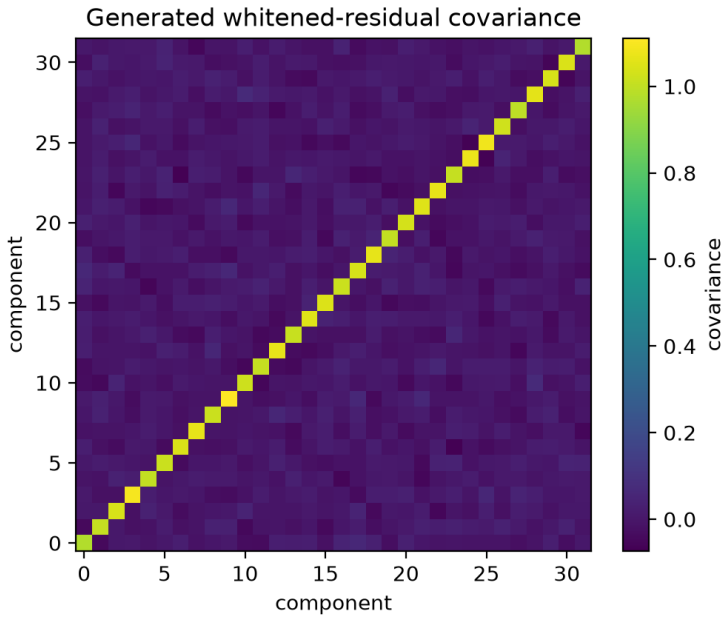


Figure 3: Generated whitened-residual covariance. Diagonality is largely recovered, but sampler bias remains in diagonal entries and small off-diagonal terms.

Table 4: Variable-coefficient Poisson warm-start pilot. The table reports the mean over four diffusion samples per condition and 128 test conditions. Relative energy error is the conductivity-weighted H^1 seminorm, and normalized residual is the interior residual RMS divided by forcing RMS.

Method	Relative L^2 ↓	Relative energy ↓	Normalized residual ↓	CPU time (s) ↓
Coarse physical initializer	0.02723	0.19572	2.97609	–
Residual DDIM	0.01962	0.03591	0.39144	7.419
Jacobi-only, 15 steps	0.02339	0.07504	0.38592	**0.00185**
Residual DDIM + operator correction	**0.01915**	**0.03070**	**0.09694**	7.522

10.2 Network and training

The denoiser uses width 48, eight dilated one-dimensional convolutional FiLM blocks, and a 96-dimensional time embedding. It is trained for 1,500 updates with batch size 64 and AdamW under a 64-step cosine DDPM; CPU wall-clock time was 278 seconds. The loss is equation (45); it adds a clean-correction loss and a diagonal-preconditioned PDE residual term with small weights to the noise MSE. Dirichlet values are hard-projected at every forward and reverse step.

Inference uses 20-step DDIM and a truncated physical start with $t_s = 0.75(T-1)$. Since $\bar{\alpha}_{t_s} = 0.144$, the initial random field is $u^{\text{init}} + 0.925 s_r \epsilon$. A Jacobi corrector with damping $\omega = 0.55$ is applied once in each of the final 15 selected steps. The fair local-solver baseline applies exactly 15 Jacobi iterations to the same coarse initializer. For each condition, the mean of four diffusion samples is compared with the deterministic target.

The table and figure below come from the preserved `legacy_reported_pilot_v0.3/` source. The v0.5 root implementation separates train, validation, test, model initialization, data loader, training noise, validation noise, evaluation sample, and trace seeds into named streams. It also generalizes the condition path with boundary lifts, $\log \beta$, deterministic/Gibbs target modes, and the residual-adaptive start in section 7.3. Those additions are source-level extensions; no new adaptive-start metrics are reported here.

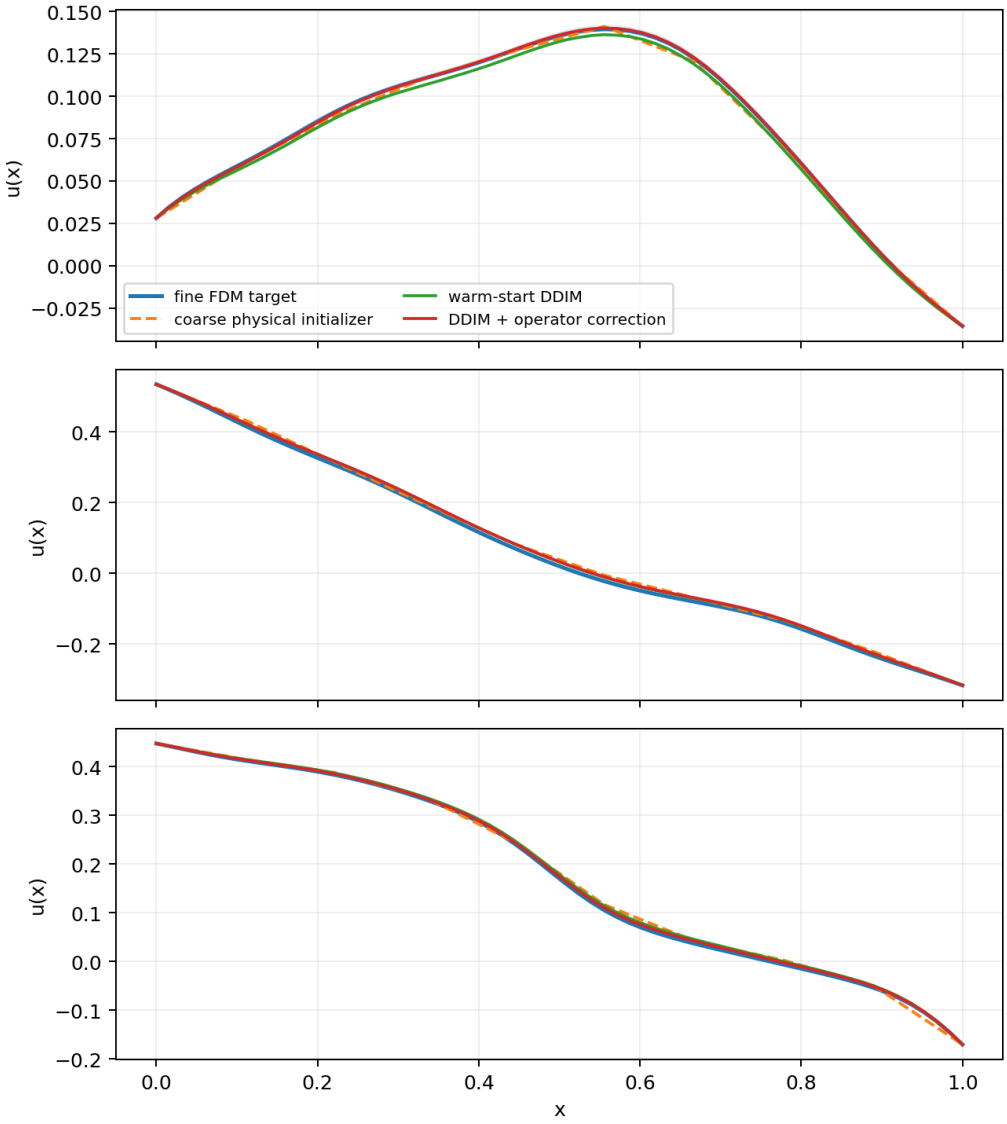


Figure 4: Variable-coefficient Poisson examples. The model denoises a correction around coarse FDM interpolation, and an operator correction is applied in the final reverse steps. Shading indicates the hybrid sample mean ±2 standard deviations; it is not a calibrated confidence interval.

10.3 Pilot results

The hybrid reduces mean relative L^2 error by 29.7%, energy error by 84.3%, and normalized residual by 96.7% compared with the coarse initializer. Compared with Jacobi-only under the same correction count, its L^2 error is 18.1% lower. Residual DDIM alone has lower L^2 and energy errors than Jacobi-only, but a similar residual; the operator corrector further reduces the residual. This is consistent with the interpretation that the learned model proposes a relatively global correction and Jacobi removes high-frequency local inconsistency.

The speed result is clearly negative. The hybrid with four samples takes 7.52 seconds, whereas 15-step Jacobi-only takes 0.00185 seconds, and a fine tridiagonal direct solve is far more appropriate for this toy problem. Thus this experiment is not evidence that a diffusion solver is faster than classical methods. It is a minimal implementation showing that a physical warm start, multi-input conditioning, hard boundary projection, and solver-native correction can be combined into one executable learning pipeline. The mean pointwise standard deviation of generated samples is 1.51×10^{-3}, but because only deterministic targets were trained, it should not be interpreted as a calibrated physical posterior.

11 Random-Geometry Generalization from Seed-Disjoint Internal-Boundary Maps

11.1 Scientific objective

The preceding Poisson experiments test exact linear Gibbs structure and one-dimensional physical-initializer feasibility. They do not test whether a learned numerical solver-error law transfers across many previously unseen physical configurations. The random-map benchmark is an operational test of low-data solver-error-law generalization. The geometries induce distinct boundary-value problems and distinct error structures for the same fixed FDM solver. The scientific endpoint is not geometry recognition by itself; the endpoint is whether SPADES can learn the solver-error law associated with one fixed lightweight FDM solver from limited physical examples and reuse that law on new problems.

A seeded random-map generator produces an internal-boundary map m_s from a seed s. In this benchmark the map changes the boundary-value problem and therefore the spatial structure of the fixed solver's error. Thus an unseen map is the operational definition of an unseen physical configuration. The random-map generator is therefore a benchmark device rather than a defining component of SPADES. Random maps are only one practical mechanism for generating physically distinct problems; a controlled family of coefficient fields, boundary conditions, forcing functions, geometries, initial states, observations, or material parameters could play the same role in a different benchmark. The present experiment varies internal boundaries while keeping other major physical parameters fixed. One fixed lightweight FDM solver gives an initializer

$$u_h^{\mathrm{init}}(m,c) = G_h(m,c), \tag{59}$$

and a higher-fidelity solver from the same numerical family gives the target $u_h^\star(m,c)$. The predicted field is the correction

$$\delta u_h(m,c) = u_h^\star(m,c) - u_h^{\mathrm{init}}(m,c). \tag{60}$$

SPADES learns a conditional denoising law

$$p_\theta\big(\delta u_h \mid u_h^{\mathrm{init}},\ R_h(u_h^{\mathrm{init}}),\ m,\ c\big)\,, \tag{61}$$

whereas the deterministic regressor learns a point map

$$\widehat{\delta u}_h = F_\phi\big(u_h^{\mathrm{init}},\ R_h(u_h^{\mathrm{init}}),\ m,\ c\big)\,. \tag{62}$$

The corrected solution is hard-projected onto the no-slip boundary constraint:

$$\widehat{u}_h = P_{\mathcal{B}_h}\Big(u_h^{\mathrm{init}} + \widehat{\delta u}_h\Big)\,. \tag{63}$$

The primary metric is correction gain,

$$G = 1 - \frac{\|\widehat{u}_h - u_h^\star\|}{\|u_h^{\mathrm{init}} - u_h^\star\|}. \tag{64}$$

Thus $G > 0$ means improvement over the initializer, $G = 0$ means no effective accuracy change, and $G < 0$ means degradation. The primary question is whether the learned solver-error law improves this same initializer on physical configurations not seen during training.

11.2 Random-map generation and fixed solvers

The benchmark uses vector-valued transient two-dimensional viscous Burgers flow on a Cartesian unit square with internal no-slip barriers. The solver evolves

$$\partial_t u + u\partial_x u + v\partial_y u = \nu\Delta u + f_u, \qquad \partial_t v + u\partial_x v + v\partial_y v = \nu\Delta v + f_v,$$

with outer and internal-barrier values set to $u = v = 0$. The implemented random map consists of ellipses and optional thin elliptical barriers rasterized on the fine grid. Each ellipse has random center, orientation, radius, and aspect ratio. The generator keeps only the largest connected fluid component and rejects maps whose interior fluid fraction is outside $[0.35, 0.93]$. It does not implement an explicit minimum-separation constraint; rejected or disconnected cases are handled by the connected-component rule. The recorded descriptors include obstacle count, solid area fraction, boundary length, connected-component count, map hash, and channel-width proxies. The current minimum-channel-width proxy is not used as a primary statistic because it is dominated by one-cell wall-adjacent distances; mean channel width is more informative in this implementation.

The fixed coarse initializer uses a 16×16 grid and 16 explicit time steps. The reference uses a 32×32 grid and 48 time steps. A medium 24×24 grid with 32 steps is generated but is not the central comparison here. The physical parameters are fixed at final time 0.18, viscosity $\nu = 0.006$, and forcing amplitude 0.35. The model receives the prolonged coarse velocity, common-grid residual, forcing, solid map, fluid mask, distance-to-wall field, coordinates, and initializer vorticity. The target has two velocity-correction channels. Corrections are masked so prescribed no-slip cells remain unchanged.

The generator ranges are those in the released configuration. Training, validation, and IID geometry use 2–4 obstacles, radius range $[0.06, 0.13]$, aspect range $[0.6, 1.8]$, and no additional thin barriers. Complexity OOD uses 5–8 smaller obstacles with aspect range $[0.45, 2.4]$. Topology OOD adds one thin barrier. Narrow-channel OOD uses 4–7 larger, more anisotropic obstacles and two thin barriers. Joint OOD combines higher obstacle count, wider radius range, stronger anisotropy, and two thin barriers.

11.3 Seed-disjoint protocol and models

The top-level seeds are

$$1,\ 2,\ 3,\ 6,\ 7,\ 14,\ 21,\ 42.$$

For each top-level seed s, the script derives disjoint seed ranges for

$$\mathcal{D}^{(s)}_{\text{train}}, \quad \mathcal{D}^{(s)}_{\text{val}}, \quad \mathcal{D}^{(s)}_{\text{test},j}.$$

Each set contains 48 maps, and each map is identified by its generator seed and a SHA-256 hash of the serialized solid mask. The audit confirms

$$\mathcal{D}^{(s)}_{\text{train}} \cap \mathcal{D}^{(s)}_{\text{val}} = \mathcal{D}^{(s)}_{\text{train}} \cap \mathcal{D}^{(s)}_{\text{test},j} = \varnothing$$

by seed and map hash for every run. Across the eight independently trained runs, the experiment therefore contains 384 training maps, 384 validation maps, 384 maps per test split, and 1,920 evaluated maps across the five test splits. These are eight independently trained models, not one model trained on all 384 training maps.

SPADES and the deterministic regressor use the same small residual CNN backbone, `FlowDenoiser` with width 32, with 133,090 trainable parameters. This is a matched-capacity comparison. The regressor directly predicts the normalized clean correction. SPADES uses x0-prediction conditional denoising diffusion: the network receives a noised correction, timestep embedding, and the same conditioning tensor, and predicts the clean correction. The diffusion schedule has 64 training steps and evaluation uses one generated sample with 32 reverse steps. Both methods use AdamW, learning rate 5×10^{-4}, batch size 8, float32, Apple MPS when available, and 5,000 updates. Training-error thresholds are recorded only as diagnostics; they do not stop training. The evaluated checkpoint is the one with the lowest mean corrected relative error on seed-disjoint validation maps.

11.4 Main results

The fixed FDM initializer has mean relative error ranging from 0.117 on IID configurations to 0.176–0.176 on narrow-channel and joint OOD cases. SPADES reduces this error in every split, with mean correction gains from 0.554 to 0.675. The deterministic regressor is a strong baseline and also improves every map, with gains from 0.534 to 0.656. Thus both learned models capture reusable information about the fixed

Table 5: Random-geometry benchmark, averaged over eight independently trained top-level seeds. Each seed uses 48 training maps, 48 validation maps, and 48 maps in each test split. Gains are defined by equation (64); positive gain means improvement over the fixed FDM initializer.

Split	Init. err.	SPADES err.	SPADES gain	Regr. err.	Regr. gain	Δ gain
IID geometry	0.1174	0.0383	0.6751	0.0405	0.6560	+0.0191
Complexity OOD	0.1371	0.0508	0.6323	0.0539	0.6098	+0.0225
Topology OOD	0.1475	0.0594	0.6015	0.0632	0.5769	+0.0245
Narrow-channel OOD	0.1762	0.0791	0.5580	0.0826	0.5381	+0.0199
Joint OOD	0.1764	0.0797	0.5539	0.0835	0.5337	+0.0202

Table 6: Seed-level paired statistics for the SPADES-minus-regressor gain difference. The independent unit is the top-level training seed ($n = 8$), not individual maps. The confidence interval is a bootstrap interval over the eight seeds, and the p-value is an exact two-sided sign-flip permutation test over 2^8 paired sign assignments.

Split	Δ mean	sample SD	bootstrap 95% CI	wins	sign-flip p
IID geometry	+0.0191	0.0114	[+0.0114,+0.0261]	8/8	0.0078
Complexity OOD	+0.0225	0.0128	[+0.0140,+0.0307]	8/8	0.0078
Topology OOD	+0.0245	0.0148	[+0.0147,+0.0338]	8/8	0.0078
Narrow-channel OOD	+0.0199	0.0153	[+0.0100,+0.0296]	7/8	0.0156
Joint OOD	+0.0202	0.0141	[+0.0121,+0.0307]	8/8	0.0078

solver's error. SPADES nevertheless exceeds the regressor in every split. The mean paired gain advantage is 0.019–0.025, with exact sign-flip $p = 0.0078$ in four splits and $p = 0.0156$ in narrow-channel OOD. The advantage is modest in absolute magnitude, but it is systematic across IID and OOD regimes.

11.5 Interpretation and limitations of the random-geometry result

This experiment indicates that the learned object is not limited to memorizing individual training problems. With only 48 training maps per independent run, SPADES transfers the solver-error law to seed-disjoint unseen physical configurations and maintains positive gains under complexity, topology, narrow-channel, and joint shifts. The random geometry is important because it changes the boundary-value problem and the spatial structure of the solver error, but geometry recognition is not the claim. The claim tested here is reusable numerical solver-error-law learning for one fixed FDM initializer.

The deterministic regressor also generalizes strongly, which demonstrates that physical-initializer correction is itself a powerful baseline. This is not a failure of deterministic regression. Rather, both models learn meaningful reusable solver-error structure from limited data, and SPADES consistently provides a modest improvement over the matched deterministic corrector under the same data regime, conditioning information, update budget, checkpoint-selection rule, and small 133k-parameter backbone. Because SPADES must also represent timestep-conditioned denoising dynamics, the matched-capacity setting is conservative for SPADES. The result supports the empirical hypothesis that diffusion-based correction may provide a better inductive bias for reusable numerical solver-error laws in this benchmark.

This paper does not establish the theoretical reason for the observed advantage. It establishes only that the effect is reproducible across repeated seed-disjoint experiments under the tested model, solver, and data regime. Possible explanations include a smoother learned correction manifold, benefits from iterative denoising, or better handling of complex residual structures induced by the random internal boundaries. These are hypotheses for future analysis, not conclusions proved by the present experiment.

The result should not be overstated. It is empirical evidence for solver-error-law transfer under one random-map generator, one Cartesian grid, one fixed Burgers FDM discretization, deterministic high-fidelity targets, and eight top-level training seeds. The generated sample variance is not a calibrated posterior. The gain over deterministic regression is consistent but modest, and practical superiority over additional classical refinement, PINNs, neural operators, or PDE-Refiner requires equal-accuracy and equal-wall-clock comparisons.

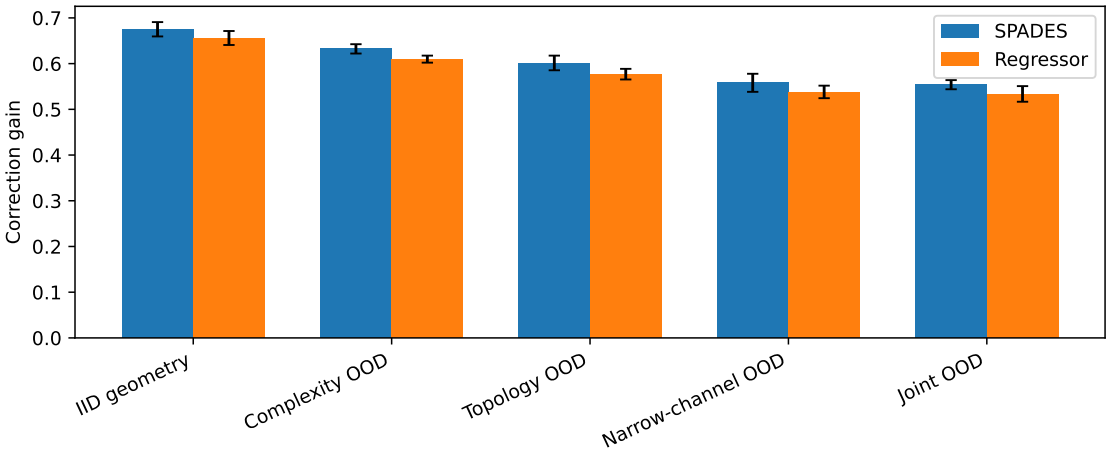


Figure 5: Correction gain by unseen random-geometry split. Error bars show sample standard deviation across the eight top-level seeds. The historical aggregate CSV used population standard deviations; this figure uses sample standard deviations for statistical reporting.

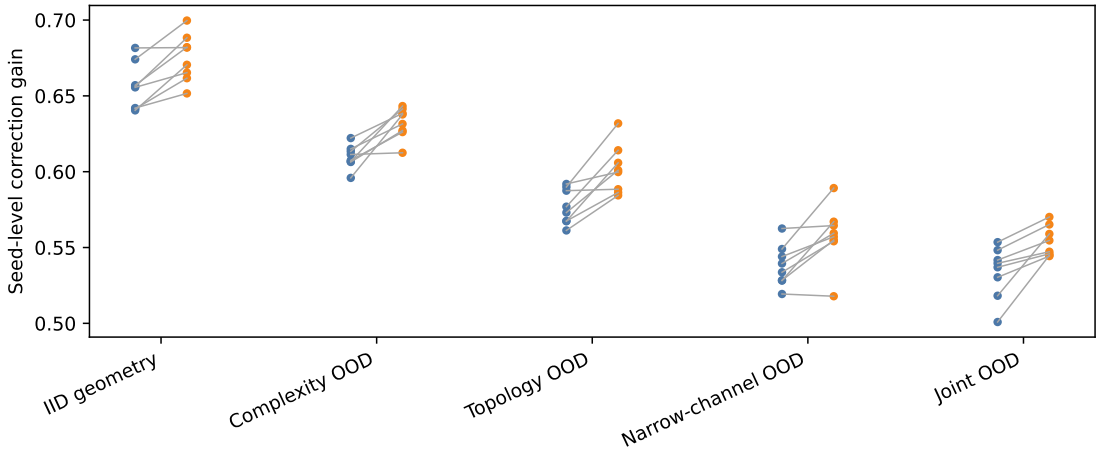


Figure 6: Paired seed-level gains. Each line connects the deterministic regressor and SPADES for one independently trained seed. SPADES wins 8/8 seeds in four splits and 7/8 seeds in the narrow-channel split.

12 Extensions

12.1 Dimension-agnostic residual-correction structure

The one-dimensional experiments are verification cases. They are not a dimensional limitation of the residual-correction formulation. Let $\Omega \subset \mathbb{R}^d$ with $d \geq 1$. A discrete problem may be written in linear form as

$$A_h^{(d)} u_h = b_h,$$

or in nonlinear form as

$$R_h^{(d)}(u_h; \xi) = 0,$$

where ξ_h collects coefficients, forcing, geometry, observations, boundary data, and other condition variables. Given a physics initializer $u_{h,\mathrm{init}}^{(d)}$, define the correction

$$\delta u_h^{(d)} = u_h^{\star,(d)} - u_{h,\mathrm{init}}^{(d)}.$$

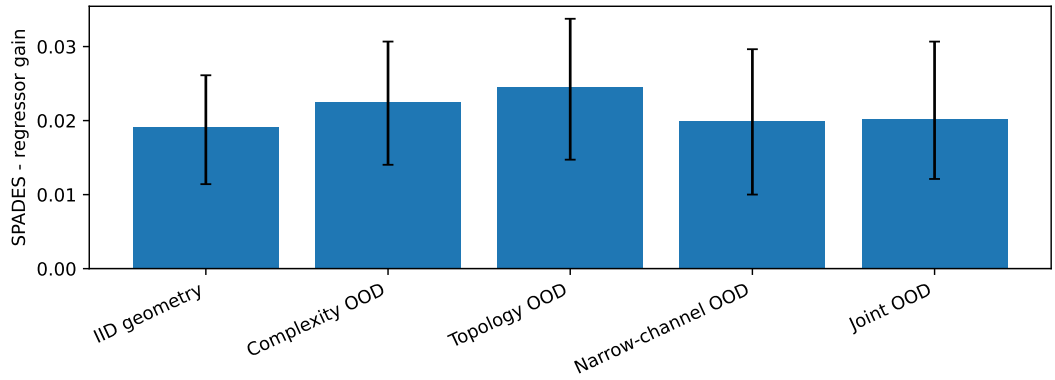


Figure 7: SPADES-minus-regressor gain difference by split. Error bars show 95% bootstrap intervals over the eight independently trained top-level seeds.

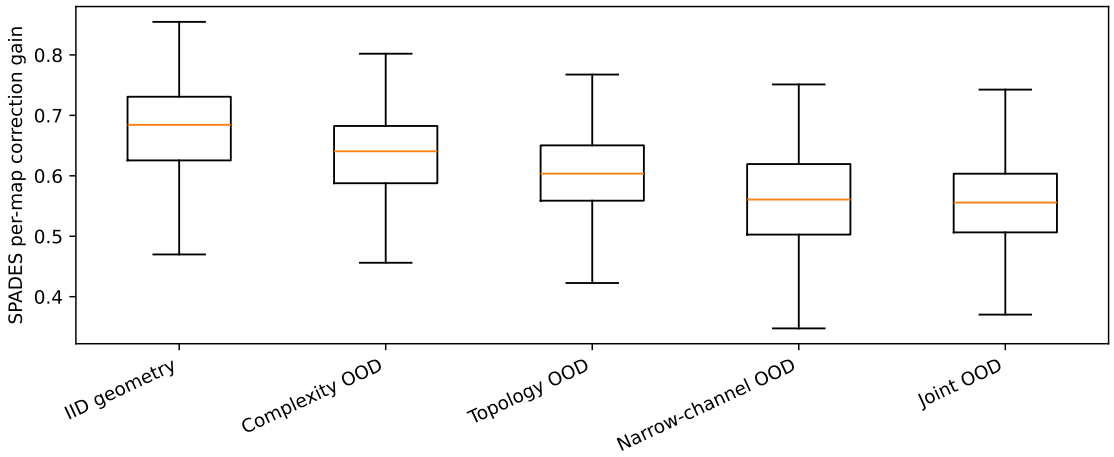


Figure 8: Per-map SPADES gain distribution over all 1,920 evaluated unseen maps. This map-level view is secondary evidence because maps are nested within trained seeds; the seed-level paired analysis in table 6 is the principal inference.

The same conditional residual-diffusion construction is

$$p_\theta\left(\delta u_h^{(d)} \mid u_{h,\mathrm{init}}^{(d)}, R_h^{(d)}(u_{h,\mathrm{init}}^{(d)}), \xi_h, \mathcal{M}_h\right). \tag{65}$$

After denoising, one reconstructs

$$\widetilde{u}_h^{(d)} = u_{h,\mathrm{init}}^{(d)} + \widehat{\delta u}_h^{(d)},$$

applies a discrete operator corrector,

$$u_{h,+}^{(d)} = \mathcal{C}_h^{(d)}\left(\widetilde{u}_h^{(d)}\right),$$

and finally imposes constraints,

$$u_{h,\mathrm{final}}^{(d)} = P_{\mathcal{B}_h^{(d)}}\left(u_{h,+}^{(d)}\right).$$

Thus the decomposition

$$\text{initializer} \to \text{correction} \to \text{denoising} \to \text{operator correction} \to \text{projection}$$

is dimension-agnostic once a consistent discrete residual and constraint projection are available.

The dimension-dependent elements are the discrete operator, mesh connectivity, boundary representation, local stencil or element assembly, neural architecture, computational cost, and memory footprint. For

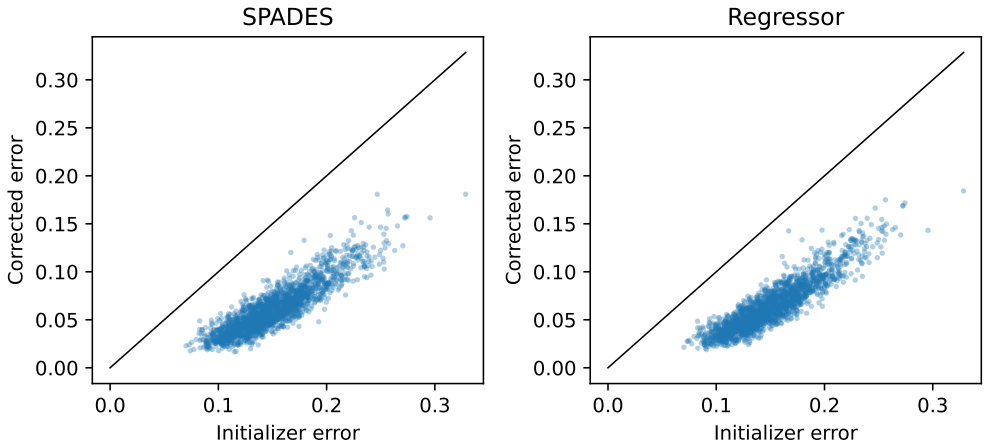


Figure 9: Initializer error versus corrected error over all seed-disjoint test maps. Points below the diagonal improve the fixed coarse FDM initializer.

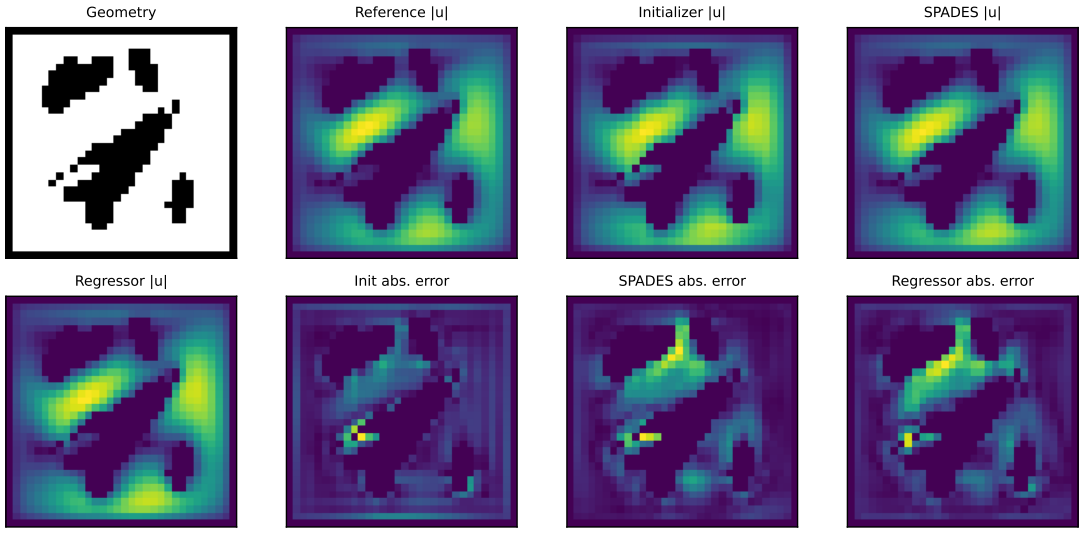


Figure 10: Representative seed-1 joint-OOD unseen map selected by a deterministic median-gain rule. The figure shows the internal-boundary map, reference speed, fixed-FDM initializer, SPADES correction, regressor correction, and absolute error fields.

Cartesian finite differences, separable elliptic operators often admit Kronecker-sum structure,

$$A_h^{(d)} = \sum_{j=1}^{d} I \otimes \cdots \otimes A_{h,j} \otimes \cdots \otimes I, \tag{66}$$

possibly modified by variable coefficients, mixed derivatives, and boundary terms. For finite elements,

$$u_h(x) = \sum_{i=1}^{N_h} c_i \phi_i(x), \tag{67}$$

and diffusion may act on the coefficient vector c, a whitened coefficient vector, or a mesh-graph field. The stiffness matrix, mass matrix, mesh graph, element features, and boundary markers can be supplied as conditioning information.

Proposition 12.1 (Dimension-independent residual formulation)**.** *For every $d \geq 1$, suppose that a stable*

discrete problem is defined by a residual map

$$R_h^{(d)} : V_h^{(d)} \to W_h^{(d)},$$

a constraint projection

$$P_{\mathcal{B}_h^{(d)}} : V_h^{(d)} \to V_{h,\mathcal{B}}^{(d)},$$

and a physics initializer

$$I_h^{(d)} : \Xi_h^{(d)} \to V_{h,\mathcal{B}}^{(d)}.$$

Then for any condition $\xi \in \Xi_h^{(d)}$, the correction variable

$$\delta u_h^{(d)} = u_h^{(d)} - I_h^{(d)}(\xi)$$

defines a residual-centered conditional probability model on the translated discrete field space. Any denoising map followed by $P_{\mathcal{B}_h^{(d)}}$ returns a constraint-admissible field.

Proof. Subtraction by the initializer is a translation of the discrete field space. By assumption, the residual $R_h^{(d)}$ is well defined in that space for the chosen dimension and discretization. A conditional probability law over $\delta u_h^{(d)}$ is therefore a law over translated discrete fields. The final projection maps the reconstructed field into $V_{h,\mathcal{B}}^{(d)}$ by definition. This proposition does not imply convergence, stability, consistency, calibration, or mesh independence; those require separate assumptions on the discretization, learned denoiser, corrector, and projection. □

Scaling caveat. Conceptual dimensional extensibility does not remove the curse of dimensionality. The number of degrees of freedom grows rapidly with d and mesh resolution; reverse diffusion may require many expensive passes; graph or sparse-operator communication increases memory traffic; and storing feature maps on high-resolution meshes can dominate computation. Practical higher-dimensional versions will likely need local, multilevel, patchwise, or latent diffusion; multigrid hierarchies; domain decomposition; sparse convolution; graph neural networks; neural operators; mesh coarsening; shortened reverse trajectories; and denoising only unresolved scales or compressed correction coordinates.

12.2 FDM

For a linear FDM system $A_h z = b_h$, using

$$E_h(z) = \frac{1}{2}(A_h z - b_h)^\mathsf{T} W_h (A_h z - b_h)$$

brings theorem 4.4 into effect. If A_h is nonsymmetric, then $H = A_h^\mathsf{T} W_h A_h$ determines the covariance geometry. The choice of W_h is important for matching physical energy. Plain identity weighting can square the conditioning of the least-squares normal equation, so preconditioning is needed.

12.3 Nonlinear FEM

The Gibbs law for a nonlinear residual $R_h(z; c) = 0$ is generally non-Gaussian. Near a minimizer z_k^* with positive-definite Hessian $H_k = \nabla^2 E_h(z_k^*)$, the Laplace approximation gives

$$\pi_{\beta,h}(z \mid c) \approx \sum_k \omega_k \mathcal{N}\big(z_k^*, (\beta H_k)^{-1}\big)\,. \tag{68}$$

Diffusion can represent this mixture, but mode coverage and the accuracy of branch weights require separate validation. Diverse branch samples can be obtained through continuation or deflation, or energy-guided MCMC can be used as a teacher.

12.4 Time-dependent PDE

For a full time-integration trajectory $z_{0:T}$, one can define an action-type energy, or define a residual Gibbs law for each one-step transition. The former has better global consistency but much higher dimension; the latter accumulates rollout error. Given the high-frequency long-horizon errors emphasized by PDE-Refiner [14], combining whitened residuals with spectral metrics may be useful.

12.5 Variable meshes and function spaces

The current code depends on a fixed mesh. Variable meshes require an architecture and representation designed for mesh transfer, such as graph neural operators, mesh-based message passing, finite-element basis coefficients, geometric multigrid features, or operator-learning constructions conditioned on K_h, M_h, and graph topology. Function-space diffusion is a separate direction aiming at discretization invariance [23, 22], and it could be combined with the mesh–temperature theory of SPADES. However, finite-dimensional Lebesgue densities cannot simply be moved to infinite dimension; one needs a reference Gaussian measure and Radon–Nikodym derivatives.

12.6 Stochastic solver-component generation

The generative object need not be restricted to the correction field. For weighted Jacobi,

$$u^{(k+1)} = u^{(k)} + \omega D^{-1}\left(b - Au^{(k)}\right),$$

a conditional generative model could produce or adapt

$$\omega, \qquad M^{-1}, \qquad P, \qquad R, \qquad S, \qquad \mathcal{C}_\theta,$$

where these symbols denote a relaxation parameter, preconditioner, prolongation operator, restriction operator, smoother, or correction operator. This would turn the broader framework into a stochastic or conditional solver-design mechanism that generates problem-dependent iterative components. Rather than learning only a solution field, one could learn a conditional distribution over solver corrections or solver components, connecting generative modeling with adaptive numerical linear algebra. This extension is not implemented or validated in the present source release.

13 Limitations and Falsifiable Research Tasks

1. **Novelty scope.** There is substantial overlap with physics-informed diffusion, CoCoGen, and function-space diffusion. The distinctiveness of the combined package "discrete Gibbs law + mesh–temperature scaling + factorized residual whitening" requires a more systematic literature review.
2. **Oracle score use.** The main experiment is a linear Gaussian theorem check and uses the exact score. This term is unavailable in nonlinear problems, and the DSM-only result shows that high-frequency calibration is difficult.
3. **Sampler bias.** The whitened covariance error 0.137 and energy ratio 1.037 show that the sampler is not exact. Matrix-valued reverse covariance, more steps, continuous-time solvers, and MALA correctors should be compared.
4. **Probability calibration.** If β is chosen arbitrarily, the distribution width is also arbitrary. A posteriori estimators, repeated-experiment coverage, empirical Bayes, or hierarchical Bayesian calibration are needed.
5. **Scalability.** The present experiments use K^{-1} or Cholesky directly and therefore do not represent large-scale problems. Sparse iterative solves, multigrid preconditioners, and local graph scores are needed.
6. **Conservation and boundary conditions.** Strong Dirichlet elimination is simple, but constraints such as incompressibility, positivity, and contact require projection or manifold diffusion.

7. **Warm-start bias and speed.** The truncated start creates the bias in equation (35), and the current hybrid pilot is thousands of times slower than Jacobi-only. Adaptive selection of the start level and equal-wall-clock comparisons are needed.
8. **Random-map benchmark scope.** The new benchmark uses one random-map generator family as a mechanism for producing varied boundary-value problems. It uses a fixed Cartesian grid, a fixed FDM discretization, deterministic high-fidelity targets, and eight top-level training seeds. The mechanism could in principle be replaced by controlled variation in coefficients, boundary data, forcing, initial states, observations, or material parameters, but those alternatives are not validated here. Topology OOD is only as broad as the implemented ellipse/barrier generator, and no FEM/FVM, variable-mesh, or cross-solver transfer is tested.
9. **Matched regressor is strong.** The deterministic correction regressor achieves gains around 0.53–0.66 on unseen maps. Because the targets are deterministic high-fidelity corrections, deterministic regression is expected to be a strong baseline. The SPADES advantage is consistent but modest. The result supports a small systematic diffusion-correction advantage for reusable solver-error-law learning in this benchmark, not failure of deterministic learned correction.
10. **Cost comparisons.** The random-geometry run used fixed 5,000-update budgets and validation-based checkpoint selection. It does not prove that diffusion is wall-clock optimal relative to larger regressors, neural operators, extra classical refinement, or solver-native correction under an equal-accuracy budget.
11. **Benchmarks.** The method should be compared with classical solvers, PINNs, neural operators, physics-informed diffusion, and MCMC on broader two-dimensional variable-coefficient elliptic problems, nonlinear elasticity, Burgers/Navier–Stokes, and multimodal bifurcations.

A clear falsification experiment is the following: under the same computational budget, if residual-Gibbs score models are consistently worse than existing methods in energy calibration, mode coverage, repeated-query wall-clock time, and out-of-distribution coefficients, then the practical value of SPADES is limited.

14 Conclusion

This work interprets the output of FDM/FEM not as a point but as a Gibbs probability measure on the discrete solution space. From this viewpoint, the classical numerical solution is the $\beta \to \infty$ limit. In linear SPD FEM, the mean $K^{-1}b$, covariance $\beta^{-1}K^{-1}$, and standard-normal whitened residual are obtained exactly. A condition coupling mesh refinement and temperature also follows.

On the learning side, the method separates the mean from the fluctuation and matches the mean and covariance of the residual instead of minimizing a plain residual. The Poisson theorem check accurately recovers the mean and temperature scaling, and score distillation substantially reduces high-frequency error hidden by coefficient-space metrics. At the same time, the remaining residual-covariance error confirms that the current sampler is not an exact probabilistic solver.

The variable-coefficient Poisson pilot shows that correction diffusion around a coarse physical solution can use several input channels to reduce initializer error, and that a classical operator corrector can further reduce the residual. It is nevertheless much slower than Jacobi-only, and its sample spread is not calibrated. The result should therefore be read as structural feasibility, not a performance advantage.

The random-map benchmark provides evidence beyond the one-dimensional pilot. With only 48 training boundary-value problems per run, SPADES transfers the solver-error law to seed-disjoint unseen physical configurations and maintains positive gains under several map-induced shifts. These experiments suggest that the learned object is not an individual solution field tied to one geometry, but a reusable solver-error law associated with a fixed numerical solver and transferable across a controlled family of unseen physical configurations. A matched deterministic regressor also generalizes strongly, demonstrating that physical-initializer correction itself is a powerful baseline when deterministic correction targets are available. SPADES nevertheless retains a small but consistent advantage across all tested regimes. The next step is to test whether this advantage survives larger problem families, higher resolution, stronger nonlinear dynamics, calibrated stochastic targets, and equal-wall-clock comparisons with classical and neural baselines.

A Proof Supplements

A.1 Quantitative bound for low-temperature concentration

Let $E_* = \min E$, and for $U \supset \mathcal{M}$ let $\Delta = \inf_{U^c} E - E_* > 0$. Choose a ball $B_r \subset U$ of radius r around a minimizer z^*. By continuity, $E \le E_* + \Delta/2$ on B_r. Then

$$\pi_\beta(U^c) \le \frac{e^{-\beta(E_*+\Delta)} \int_{U^c} e^{-\beta(E-E_*-\Delta)} \mathrm{d}z}{|B_r| e^{-\beta(E_*+\Delta/2)}} \le C_\beta e^{-\beta\Delta/2},$$

where coercivity controls C_β. Hence the probability converges to zero exponentially.

A.2 Cross term in the mesh–temperature identity

Let $e_h = u - u_h^*$ and $\delta_h = U_h - u_h^*$. Then

$$\mathbb{E}\,\|u - U_h\|_a^2 = \|e_h\|_a^2 + \mathbb{E}\,\|\delta_h\|_a^2 - 2\mathbb{E}a(e_h, \delta_h).$$

Because $\mathbb{E}\delta_h = 0$, the last term is zero. Galerkin orthogonality is not needed for this cross term, although the fact that u_h^* is the Galerkin projection enters the deterministic error bound.

B Core Implementation Excerpts

The full code is included in the reproduction package. The following snippets show the target distribution and temperature separation.

Listing 1: Exact FEM Gibbs sample and temperature separation.

```
def gibbs_sample(rng, means, cholesky, beta):
    xi = rng.normal(size=means.shape)
    delta = np.linalg.solve(
        cholesky.T,
        (xi / np.sqrt(beta)[:, None]).T,
    ).T
    return means + delta

# v is independent of beta for the linear SPD Gibbs family.
v = np.sqrt(beta)[:, None] * (target - deterministic_mean)
```

Listing 2: Physics-informed fluctuation objective.

```
noise = torch.randn_like(v0)
v_t = diffusion.q_sample(v0, t, noise)
eps_pred = denoiser(v_t, t)
dsm = F.mse_loss(eps_pred, noise)

v0_pred = diffusion.predict_x0(v_t, t, eps_pred).clamp(-6, 6)
v_phys = v0_pred * fluctuation_std
whitened = v_phys @ cholesky       # y = L^T v for row vectors
cov = torch.cov(whitened.T)
moment = whitened.mean(0).square().mean()
moment += (cov - torch.eye(n_dof)).square().mean()

score_distill = F.mse_loss(denoiser.weight, exact_W_t)
loss = dsm + 0.01 * moment + 100.0 * score_distill
```

Listing 3: Physical warm-start residual diffusion and solver-native correction.

```
# The network never denoises the full field from white noise.
# It denoises a normalized correction around a coarse physical solve.
r0 = (u_fine - u_init) / residual_scale
condition = torch.cat([
```

```
    u_init, conductivity, forcing, initial_pde_residual,
    coordinate, left_bc_field, right_bc_field, interior_mask,
], dim=1)

noise = torch.randn_like(r0) * interior_mask
r_t = diffusion.q_sample(r0, t, noise)
eps = model(r_t, t, condition, diffusion.n_steps)
r0_hat = diffusion.predict_x0(r_t, t, eps) * interior_mask
u_hat = project_dirichlet(u_init + residual_scale * r0_hat, bc)

# Optional FDM/FEM corrector inside late reverse steps.
u_hat = jacobi_correct(
    u_hat, conductivity, forcing, bc, dx, omega=0.55
)
r0_hat = (u_hat - u_init) / residual_scale
```

Listing 4: Residual-adaptive reverse start in the v0.5 source.

```
# q_proxy is the RMS normalized correction estimated by a cheap
# weighted-Jacobi preconditioner. rho_t is the schedule noise/signal ratio.
q_proxy = normalized_reference_error_proxy(
    reference, coefficient, forcing, boundary,
    correction_scale,
    estimator="jacobi", jacobi_steps=8,
)
rho = torch.sqrt((1.0 - diffusion.alpha_bar) / diffusion.alpha_bar)
target = 2.0 * q_proxy[:, None]  # kappa = 2 safety factor
start_steps = candidate_steps[
    (torch.log(rho[candidate_steps])[None] - torch.log(target)).abs().argmin(1)
]
field, trace = reference_centered_ddim_sample(
    diffusion, model, condition, reference, correction_scale,
    generator=cpu_generator(271828),
    start_steps=start_steps,
)
```

C Reproduction Commands and Fixed Seeds

The v0.5 bundle is a source-only handoff. It includes both the preserved source for the numerical table already reported in this manuscript and a newer generalized conditioned source path. All current seeds are recorded in `configs/seeds.json` and `svd_pde/seeds.py`. For strict local reproduction, `PYTHONHASHSEED` should be set before Python starts.

Table 7: Independent random streams in the current root implementation. The streams are separated so that changes in one subsystem do not silently shift the random sequence of another.

Stream	Seed	Stream	Seed
Exact dataset	17011	Conditioned train data	31011
Exact mean model	17012	Conditioned validation data	31012
Exact mean loader	17013	Conditioned test data	31013
Exact fluctuation model	17014	Conditioned model	31014
Exact fluctuation loader	17015	Conditioned loader	31015
Exact training noise/time	17016	Conditioned training noise/time	31016
Exact evaluation	31415	Conditioned validation noise/time	31017
FEM unit test	41011	Conditioned evaluation	271828
Factorization unit test	41012	Conditioned path trace	161803
Conditioned-data unit test	41013	Conditioned-sampling unit test	41014

Install and static tests.

```
python -m pip install -r requirements.txt
python -m pip install -e .
pytest -q
```

Linear FEM Gibbs theorem check.

```
PYTHONHASHSEED=17012 python train.py \
  --output results/factorized_seeded \
  --dataset-seed 17011 \
  --mean-model-seed 17012 --mean-loader-seed 17013 \
  --fluctuation-model-seed 17014 --fluctuation-loader-seed 17015 \
  --training-noise-seed 17016

PYTHONHASHSEED=31415 python evaluate.py \
  results/factorized_seeded/model.pt --sampling-seed 31415
```

Current conditioned source path.

```
PYTHONHASHSEED=31014 python train_conditioned.py \
  --output results/conditioned_seeded --target-law deterministic \
  --train-data-seed 31011 --validation-data-seed 31012 \
  --test-data-seed 31013 --model-seed 31014 --loader-seed 31015 \
  --training-noise-seed 31016 --validation-noise-seed 31017

# Fixed physical start, comparable in spirit to the reported pilot.
PYTHONHASHSEED=271828 python evaluate_conditioned.py \
  results/conditioned_seeded/conditioned_model.pt \
  --test-data results/conditioned_seeded/conditioned_test_data.npz \
  --start-mode fixed --start-fraction 0.75 \
  --sampling-seed 271828 --trace-seed 161803

# Residual-adaptive start implemented in v0.5. No paper metrics are reported.
PYTHONHASHSEED=271828 python evaluate_conditioned.py \
  results/conditioned_seeded/conditioned_model.pt \
  --test-data results/conditioned_seeded/conditioned_test_data.npz \
  --start-mode residual-adaptive --adaptive-estimator jacobi \
  --adaptive-jacobi-steps 8 --adaptive-jacobi-relaxation 0.65 \
  --adaptive-min-fraction 0.05 --adaptive-max-fraction 0.95 \
  --adaptive-safety-factor 2.0 \
  --sampling-seed 271828 --trace-seed 161803
```

Archived reported warm-start pilot.

```
cd legacy_reported_pilot_v0.3
PYTHONHASHSEED=29 python train_warmstart.py \
  --output results/warmstart_reported --train-steps 1500 --seed 29
PYTHONHASHSEED=29 python evaluate_warmstart.py \
  results/warmstart_reported/model.pt \
  --samples 4 --sampling-steps 20 --start-fraction 0.75 \
  --physics-corrections 1 --limit 128
```

Random-geometry seed-disjoint benchmark. The following commands reproduce the benchmark in section 11. They use 48 maps in train, validation, and each test split for every top-level seed. Training runs for the full 5,000 updates; model selection uses seed-disjoint validation corrected error.

```
cd SPADES_Condition_Generalization_Benchmark

# One-seed debug run.
CONFIG=configs/random_flow/trainfit_005_48test.json \
RESULTS=results/random_flow/seed_sweep_generalization_48test_seed1 \
SEEDS=1 \
./scripts/run_random_flow_seed_sweep.sh

# Full eight-seed run used in this manuscript.
CONFIG=configs/random_flow/trainfit_005_48test.json \
RESULTS=results/random_flow/seed_sweep_generalization_48test \
SEEDS="1 2 3 6 7 14 21 42" \
./scripts/run_random_flow_seed_sweep.sh

cd ..

# Recompute tables, figures, checksums, audit, and statistics from raw files.
```

```
MPLCONFIGDIR=.matplotlib-cache \
.venv/bin/python scripts/analyze_random_geometry_results.py

# Build the manuscript.
cd paper
pdflatex -interaction=nonstopmode -halt-on-error main.tex
pdflatex -interaction=nonstopmode -halt-on-error main.tex
```

CPU/GPU, BLAS, PyTorch, and compiler versions can change floating-point results even when deterministic algorithms are requested. Checkpoint configs, package versions, device, dtype, and the seed manifest should therefore be recorded together. The archived commands reproduce the historical path used before v0.5 seed separation. The current root scripts should be preferred for new runs.

D Notation

Symbol	Meaning
h, n_h	mesh size and number of discrete degrees of freedom
z, z_h^*	stochastic discrete state and deterministic discrete solution
R_h, W_h	discrete residual and residual metric
E_h, Φ_h	Gibbs energy and additional potential
β	inverse temperature; larger values concentrate near the deterministic solution
K, b, L	FEM stiffness, load, and Cholesky factor $K = LL^\mathsf{T}$
v	temperature-separated fluctuation $\sqrt{\beta}(z - z_h^*)$
y	whitened residual $L^\mathsf{T} v$
$\bar{\alpha}_t$	diffusion cumulative signal coefficient
ρ_t, κ	diffusion noise-to-signal ratio and adaptive-start safety factor
W_t^*	optimal noise-prediction matrix for the Gaussian noised marginal

References

[1] Assyr Abdulle and Giacomo Garegnani. A probabilistic finite element method based on random meshes: A posteriori error estimators and Bayesian inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 384:113961, 2021.

[2] 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, July 2019.

[3] Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *International Conference on Learning Representations*, volume 2025, pages 3360–3385, 2025.

[4] Jon Cockayne, Chris Oates, Tim Sullivan, and Mark Girolami. Probabilistic numerical methods for PDE-constrained Bayesian inverse problems. In *AIP Conference Proceedings*, volume 1853, page 060001. AIP Publishing LLC, 2017.

[5] Jon Cockayne, Chris J. Oates, T. J. Sullivan, and Mark Girolami. Bayesian Probabilistic Numerical Methods. *SIAM Review*, 61(3):756–789, January 2019.

[6] Patrick R. Conrad, Mark Girolami, Simo Särkkä, Andrew Stuart, and Konstantinos Zygalakis. Statistical analysis of differential equations: Introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082, July 2017.

[7] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *International Conference on Machine Learning*, pages 2415–2423. PMLR, 2019.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[9] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-solving under partial observation. *Advances in Neural Information Processing Systems*, 37:130291–130323, 2024.

[10] Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. CoCoGen: Physically Consistent and Conditioned Score-Based Generative Models for Forward and Inverse Problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, April 2025.

[11] 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, May 2021.

[12] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[13] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021.

[14] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

[15] 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.

[16] Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh. Learning algebraic multigrid using graph neural networks. In *International Conference on Machine Learning*, pages 6489–6499. PMLR, 2020.

[17] Jaewan Park, Farid Ahmed, Kazuma Kobayashi, Seid Koric, Syed Bahauddin Alam, Iwona Jasiuk, and Diab Abueidda. Bridging Sequential Deep Operator Network and Video Diffusion: Residual Refinement of Spatio-Temporal PDE Solutions, July 2025.

[18] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[19] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José M. Hernández-Lobato, Richard E. Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. *Advances in Neural Information Processing Systems*, 37:23246–23300, 2024.

[20] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021.

[21] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in neural information processing systems*, 33:6111–6122, 2020.

[22] Sifan Wang, Zehao Dou, Siming Shan, Tong-Rui Liu, and Lu Lu. Fundiff: Diffusion models over function spaces for physics-informed generative modeling. *Nature Communications*, 2026.

[23] Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Animashree Anandkumar. Guided diffusion sampling on function spaces with applications to pdes. *Advances in Neural Information Processing Systems*, 38:127057–127094, 2026.

[24] Yi Zhang, Peng Wang, and Difan Zou. Physics-Informed Distillation of Diffusion Models for PDE-Constrained Generation, June 2026.