SCaSML
Simulation-Calibrated
Scientific Machine Learning

Physics-Informed Inference Time Scaling via Simulation-Calibrated Scientific Machine Learning

Zexi Fan[1], Yan Sun [2], Shihao Yang[3], Yiping Lu*[4]

[1] **Peking University** [2] **Visa Inc.** [3] **Georgia Institute of Technology** [4] **Northwestern University**
`fanzexi_francis@stu.pku.edu.cn,yansun414@gmail.com,`
`shihao.yang@isye.gatech.edu,yiping.lu@northwestern.edu`

Abstract

High-dimensional partial differential equations (PDEs) pose significant computational challenges across fields ranging from quantum chemistry to economics and finance. Although scientific machine learning (SciML) techniques offer approximate solutions, they often suffer from bias and neglect crucial physical insights. Inspired by inference-time scaling strategies in language models, we propose Simulation-Calibrated Scientific Machine Learning (SCaSML), a physics-informed framework that dynamically refines and debiases the SCiML predictions during inference by enforcing the physical laws. SCaSML leverages derived new physical laws that quantifies systematic errors and employs Monte Carlo solvers based on the Feynman–Kac and Elworthy–Bismut–Li formulas to dynamically correct the prediction. Both numerical and theoretical analysis confirms enhanced convergence rates via compute-optimal inference methods. Our numerical experiments demonstrate that SCaSML reduces errors by 20–50% compared to the base surrogate model, establishing it as the first algorithm to refine approximated solutions to high-dimensional PDE during inference. Code of SCaSML is available at `https://github.com/Francis-Fan-create/SCaSML`.

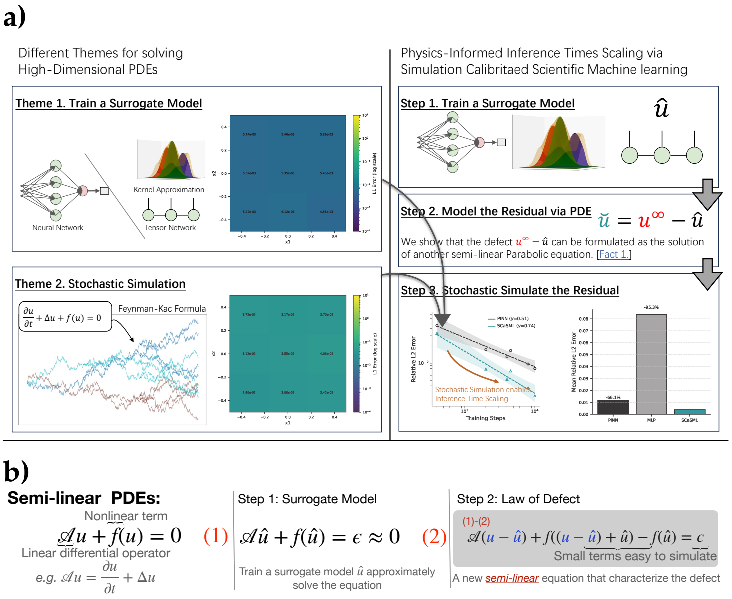

a)
Different Themes for solving High-Dimensional PDEs
Theme 1. Train a Surrogate Model
Neural Network
Kernel Approximation
Tensor Network
5.14e-03
5.46e-03
5.39e-03
5.02e-03
5.35e-03
5.43e-03
5.72e-03
6.12e-03
4.56e-03
x1
x2
L1 Error (log scale)
Theme 2. Stochastic Simulation
$\frac{\partial u}{\partial t} + \Delta u + f(u) = 0$
Feynman-Kac Formula
3.72e-02
3.17e-02
3.70e-02
3.12e-02
3.55e-02
4.03e-02
2.80e-02
3.08e-02
3.42e-02
x1
x2
L1 Error (log scale)
Physics-Informed Inference Times Scaling via Simulation Calibritaed Scientific Machine learning
Step 1. Train a Surrogate Model
$\hat{u}$
Step 2. Model the Residual via PDE
$\breve{u} = u^\infty - \hat{u}$
We show that the defect $u^\infty - \hat{u}$ can be formulated as the solution of another semi-linear Parabolic equation. [Fact 1.]
Step 3. Stochastic Simulate the Residual
PINN (γ=0.51)
SCaSML (γ=0.74)
Relative L2 Error
Training Steps
Stochastic Simulation enables Inference Time Scaling
Mean Relative L2 Error
-66.1%
-95.3%
PINN
MLP
SCaSML
b)
Semi-linear PDEs:
Nonlinear term
$\underbrace{\mathscr{A}}u + \overbrace{f}(u) = 0$ (1)
Linear differential operator
e.g. $\mathscr{A}u = \frac{\partial u}{\partial t} + \Delta u$
Step 1: Surrogate Model
$\mathscr{A}\hat{u} + f(\hat{u}) = \epsilon \approx 0$ (2)
Train a surrogate model $\hat{u}$ approximately solve the equation
Step 2: Law of Defect
(1)-(2)
$\mathscr{A}(u - \hat{u}) + f((u - \underbrace{\hat{u}) + \hat{u}) - f(\hat{u})} = \underbrace{\epsilon}$
Small terms easy to simulate
A new semi-linear equation that characterize the defect

Figure 1 | **SCaSML framework pipeline. a)** SCaSML aims to allocate compute at inference time to further improve the accuracy of a surrogate model. It first fits a surrogate model $\hat{u}$ as an initial estimate of the PDE solution u^∞, then leverages stochastic simulation algorithms to approximate the `defect` $\breve{u} = u^\infty - \hat{u}$ at inference time via formulating $\breve{u}$ as the solution to the law of defect. **b)** Method for deriving the law of defect. Using an approximate solution of a semi-linear equation, we derive a new differential equation that characterizes the error, which inherently preserves the semi-linear structure.

* Corresponding Author.

Contents

1. Introduction

Accurate predictions of complex physical systems often rely on simulating partial differential equations (PDEs), but high-dimensional PDEs pose significant computational challenges. Traditional methods, such as numerical simulations and analytical solutions, can be computationally expensive and limited in scope. Recently, scientific machine learning (SciML) (Brunton and Kutz, 2024; Han et al., 2018a; Long et al., 2018, 2019; Raissi et al., 2017), has emerged as a promising approach that combines data-driven models with fundamental physical principles to enhance both accuracy and efficiency in high-dimensional PDE simulations. However, SCiML models can be biased due to limitations in their training processes, which may compromise prediction accuracy. These biases raise concerns about their reliability compared to traditional, theory-backed scientific computing methods.

Recent advances in large language models (LLMs) have shown that inference-time scaling, using extra computational resources during prediction, can significantly improve output quality (Brown et al., 2024; Gandhi et al., 2024; Snell et al., 2024; Wei et al., 2022; Wu et al., 2024). In this work, we investigate the impact of inference-time scaling for SCiML models and answers the following question:

How can we refine a machine-learned PDE solver at inference time to further improve accuracy and reliability?

We introduce a novel physics-informed, inference-time scaling framework for scientific machine learning. Our work aims to addresses high-dimensional partial differential equations afflicted by the curse of dimensionality, such as the Schrödinger equation in quantum many-body systems, the nonlinear Black–Scholes equation in financial derivatives pricing, and the Hamilton–Jacobi–Bellman equation in dynamic programming. Notably, these semi-linear parabolic PDEs (Han et al., 2018a; Hutzenthaler et al., 2019; Weinan et al., 2021) can be expressed as:

$$\begin{cases} \frac{\partial}{\partial r}u^\infty + \left\langle \mu, \nabla_y u^\infty \right\rangle + \frac{1}{2}\mathrm{Tr}(\sigma^* \mathrm{Hess}\, u^\infty\ \sigma) + F(u^\infty, \sigma^* \nabla_y u^\infty) = 0, \text{ on } [0,T) \times \mathbb{R}^d \\ u^\infty(T,y) = g(y), \text{ on } \mathbb{R}^d. \end{cases} \tag{1}$$

where $T > 0$, $d \in \mathbb{N}$, $g : \mathbb{R}^d \to \mathbb{R}$, $u^\infty : [0,T] \times \mathbb{R}^d \to \mathbb{R}$, $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, and σ is a regular function mapping $[0,T] \times \mathbb{R}^d$ to an invertible $d \times d$ real matrix. We then propose Simulation-Calibrated Scientific Machine Learning (SCaSML), which calibrates the SCiML model during inference. SCaSML first trains a surrogate model $\hat{u}$—which may be constructed using sparse grid (Bungartz and Griebel, 2004), neural networks (Han et al., 2018a; Karniadakis et al., 2021; Yu et al., 2022), tensor networks (Bachmayr, 2023; Bachmayr et al., 2016; Chen et al., 2023; Richter et al., 2021), or Gaussian processes (Chen et al., 2021; Yang et al., 2021)—and subsequently introduces a novel `Law of Defect` (Figure 1 b) and Fact 1), a new partial differential equation (PDE) where $u - \hat{u}$ represents its solution, aimed at quantifying the defect $u - \hat{u}$. Remarkably, the `Law of Defect` inherits the semilinear structure, enabling a stochastic simulation algorithm to correct errors via Feynman path-based stochastic simulation algorithms.

Section 4 shows that using a pre-trained surrogate confines error accumulation to discrepancies along the Feynman path. A better surrogate reduces simulation variance during inference, thereby improving the convergence rate (see Remark 1). Moreover, our numerical experiments demonstrate that inference-time corrections scale across broad solution domains for various high-dimensional PDEs, reducing errors by approximately 50–80% for naive stochastic simulation algorithms and 10–50% for surrogate models. The main contributions of this work can be summarized as follows:

- We introduce the first (physics-informed) inference-time scaling framework for scientific machine learning via derive a novel `Law of Defect` that quantifies the error in the learned surrogate model. The `Law of Defect` inherent the semi-linear structure which enables the use of a stochastic simulation algorithm to correct the surrogate model by simulating error propagation along the Feynman path.
- We theoretically demonstrate that the error of the SCaSML algorithm can be bounded by the product of the error of the stochastic simulation algorithm and the error from the surrogate model, resulting in an improved convergence rate.
- We numerically evaluate SCaSML's inference-time scaling for high-dimensional PDEs. Applied to Gaussian Process and physics-informed neural network surrogates, SCaSML reduces errors by $20 \sim 50\%$ for 100 dimensioanl PDEs, highlighting its promise in overcoming the curse of dimensionality.

1.1. Related Works and Preliminaries

Hybrid Paradigm Hybrid approaches combining scientific computing and machine learning have been explored—such as learning numerical schemes (Long et al., 2018, 2019) and iterative method preconditioners (Hsieh et al., 2019; Zhang et al., 2022). However, SCaSML is the first framework to achieve faster convergence than both traditional numerical solvers and machine learning surrogates. Unlike recent studies (Suo and Zhang, 2023; Zhou et al., 2023) that debias solutions over the entire domain using finite difference techniques, SCaSML leverages a Monte Carlo algorithm to refine a single state-space solution, yielding rapid convergence improvements. In contrast to learning the PDE solution in whole-domain, whose convergence rate is theoretically optimal using SCiML models (Lu et al., 2022a,b; Nickl et al., 2020), our targeted debiasing strategy enables further convergence gains.

High-Dimensional PDE Solvers Many fundamental problems in science and engineering are modeled using high-dimensional partial differential equations (PDEs). Examples include:

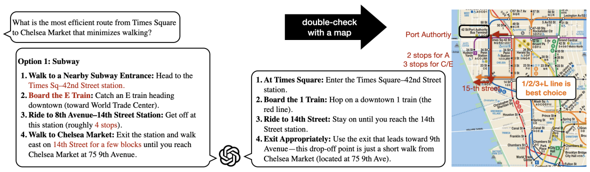
- Schrödinger equation in quantum many-body systems, where the dimension scales with the number of particles.

- Nonlinear Black–Scholes equation for pricing financial derivatives, with dimensionality depending on the number of assets.
- Hamilton–Jacobi–Bellman equation in dynamic programming, where the dimension grows with the number of agents or resources.

Notably, all these equations are semilinear parabolic PDEs (Han et al., 2018a; Hutzenthaler et al., 2019; Weinan et al., 2021) where Monte Carlo systems are being explored to overcome the "curse of dimension", such as BSDE-based methods with discretized expectations(Briand and Labart, 2014; Geiss and Labart, 2016), branching diffusion methods (Bouchard et al., 2017; Henry-Labordère et al., 2019), and the full-history multilevel Picard Iteration (MLP) (Grohs et al., 2022; Hutzenthaler et al., 2020a). Neural networks in BSDE handle a variety of semilinear equations but presents significant challenges due to both extended training durations and sensitivity to parameters(Beck et al., 2019). Branching diffusion techniques are rather effective in the case of certain nonlinear PDEs, yet its error analysis only applies to limited time scope and small initial condition(Henry-Labordère et al., 2019). MLP alleviate the computational challenges associated with high-dimensional gradient-independent PDEs (Hutzenthaler et al., 2021), theoretically achieving a linear rate of convergence in relation to dimensionality, across any time frame.

2. Warm Up:High-Dimensional Linear Parabolic Equations

> Consider the scenario of organizing a trip wherein an AI assistant proposes a route, but you corroborate it through a map. Likewise, in this study, we enhance the predictions of neural networks employing a Feynman path simulation approach.



What is the most efficient route from Times Square to Chelsea Market that minimizes walking?

Option 1: Subway

1. **Walk to a Nearby Subway Entrance:** Head to the Times Sq–42nd Street station.
2. **Board the E Train:** Catch an E train heading downtown (toward World Trade Center).
3. **Ride to 8th Avenue–14th Street Station:** Get off at this station (roughly 4 stops).
4. **Walk to Chelsea Market:** Exit the station and walk east on 14th Street for a few blocks until you reach Chelsea Market at 75 9th Avenue.

1. **At Times Square:** Enter the Times Square–42nd Street station.
2. **Board the 1 Train:** Hop on a downtown 1 train (the red line).
3. **Ride to 14th Street:** Stay on until you reach the 14th Street station.
4. **Exit Appropriately:** Use the exit that leads toward 9th Avenue—this drop-off point is just a short walk from Chelsea Market (located at 75 9th Ave).

Directly learning solutions for partial differential equations (PDEs) with a neural network is often unreliable and difficult to validate because of inherent approximation errors and a lack of convergence guarantees. To address these challenges, we propose a novel **S**imulation-**Ca**librated **S**cientific **M**achine **L**earning (SCaSML) framework. First, the neural network generates an initial, approximate solution that serves as a guiding estimate. We then refine this estimate by incorporating the underlying physical principles. Specifically, we derive an equation governing the discrepancy, $\hat{u} - u^\infty$, where $\hat{u}$ represents the neural network solution and u^∞ denotes the true solution. We term this equation the "Law of Defect" and analyze its behavior using stochastic simulation methods to enhance the reliability and robustness of the final solution.

This section presents our methodology through linear parabolic equations—a simplified example that elucidates both our underlying philosophy and the rationale for achieving enhanced convergence rates. Consider the high-dimensional linear parabolic PDE:

$$\begin{cases} \frac{\partial}{\partial r} u(r,y) + \left\langle \mu, \nabla_y u(r,y) \right\rangle + \frac{1}{2} \mathrm{Tr}\Big(\sigma^* \, \mathrm{Hess}_x \, u(r,y) \, \sigma\Big) = f(r,y), & \text{on } [0,T) \times \mathbb{R}^d, \\ u(T,y) = g(y), & \text{for all } y \in \mathbb{R}^d. \end{cases} \tag{2}$$

To address the challenges posed by high dimensionality, surrogate models such as Gaussian processes (Chen et al., 2021; Yang et al., 2021), tensor networks (Bachmayr, 2023; Bachmayr et al., 2016; Chen et al., 2023), and neural networks (E and Yu, 2018; Raissi et al., 2017; Sirignano and Spiliopoulos, 2018) are typically employed to optimize a variational formulation over randomly sampled test points. However, these approaches often suffer from convergence issues and overfitting in high-dimensional settings. The SCaSML framework overcomes these challenges by correcting the biased solution $\hat{u}$ through inference-time scaling. This is accomplished by deriving a new PDE—referred to as the `Law of defect`—which, when integrated into the surrogate model, effectively rectifies its shortcomings. Additionally, a numerical simulator is employed at inference time to pinpoint the physical mechanisms underlying the bias.

Derivation of `Law of Defect` We begin by defining the residual of the surrogate model $\hat{u}$ with respect to the given partial differential equation (PDE). Specifically, we define

$$\epsilon(r,y) := f(r,y) - \left(\frac{\partial \hat{u}(r,y)}{\partial r} + \langle \mu, \nabla_y \hat{u}(r,y)\rangle + \frac{1}{2}\operatorname{Tr}\Big(\sigma^* \operatorname{Hess}_y \hat{u}(r,y)\, \sigma\Big)\right),$$

which quantifies the discrepancy between the learned surrogate model $\hat{u}$ and the PDE (2). If we denote the defect by $\breve{u}(r,y) := u(r,y) - \hat{u}(r,y)$, then the defect satisfies what we refer to as the `Law of Defect`.

The $\breve{u}(r,y) := u(r,y) - \hat{u}(r,y)$ is the solution to the following linear Parabolic equation:

$$\begin{cases} \dfrac{\partial \breve{u}(r,y)}{\partial r} + \langle \mu, \nabla_y \breve{u}(r,y)\rangle + \dfrac{1}{2}\operatorname{Tr}\Big(\sigma^* \operatorname{Hess}_y \breve{u}(r,y)\, \sigma\Big) = \epsilon(r,y), & \text{on } [0,T)\times\mathbb{R}^d, \\ \breve{u}(T,y) = g(y) - \hat{u}(T,y), & \text{for all } y \in \mathbb{R}^d, \end{cases} \tag{3}$$

which we call it `Law of Defect`.

If an estimated solution to `Law of defect` is obtained, the approximated defect resulting can be used to refine the output of the learned surrogate model.

Rectify the Answer by Correcting Errors Along the Feynman Path To estimate the solution of the law of defect, we can use the Feynman–Kac formula to express the defect $\breve{u}(s,x)$ as the expectation of the error along the Feynman path. Specifically, we have

$$\breve{u}(s,x) = \mathbb{E}\Big[g\big(X_T^{s,x}\big) - \hat{u}\big(T, X_T^{s,x}\big)\Big] + \underbrace{\mathbb{E}\Big[\int_s^T \epsilon\big(t, X_t^{s,x}\big)dt\Big]}_{\text{accumulated error along the Feynman path}}. \tag{4}$$

By simulating the Feynman path and collecting the error ϵ as indicated in (4), one can construct an unbiased stochastic simulation for the defect $\breve{u}$.

Remark 1 (How Does SCaSML Achieve a Faster Convergence Rate?). *By employing a pre-trained surrogate, the SCaSML algorithm accumulates only the error $\epsilon(t, X_t^{s,x})$ along the Feynman path. Since the simulation variance is proportional to $\epsilon(t, X_t^{s,x})$, improving surrogate accuracy reduces the variance. If an algorithm provides a surrogate model with error $\epsilon(t, X_t^{s,x}) \sim m^{-\gamma}$ using m collocation points, the SCaSML simulation achieves a variance of order $m^{-2\gamma}$. With an additional m random realizations, the final error is reduced to order $\frac{\sqrt{m^{-2\gamma}}}{\sqrt{m}} = m^{-\frac{1}{2}-\gamma}$. Thus, with $2m$ collocation points, SCaSML attains an accelerated convergence rate of $m^{-\frac{1}{2}-\gamma}$, outperforming both the Monte Carlo algorithm, which converges at $m^{-\frac{1}{2}}$, and the original surrogate model, which converges at $m^{-\gamma}$.*

Remark 2 (The Reason for Using Monte Carlo in the Debiasing Step)**.** *The SCaSML framework comprises two steps. First, a surrogate model approximates the solution of a partial differential equation (PDE); then, a rigorous solver refines this surrogate. We use the Monte Carlo algorithm for debiasing because it effectively manages variance regardless of input regularity .*

Learning theory shows that surrogates initially capture smooth function components, leading to oscillatory, irregular errors. In contrast to other models based on approximation theory—which require function regularity—the Monte Carlo method does not. Our analysis confirms that the initial estimator reduces variance, highlighting the synergy and effectiveness of our approach.

2.1. Numerical Experiments

We investigate a linear convection-diffusion equation given by

$$\frac{\partial}{\partial r}u^{\infty}(r,y)+\left\langle -\frac{1}{d}\mathbf{1},\nabla_y u^{\infty}(r,y)\right\rangle+\Delta_y u^{\infty}(r,y)=0,\quad (r,y)\in[0,T)\times\mathbb{R}^d, \tag{5}$$

with terminal condition $u^{\infty}(T,y)=\sum_{i=1}^d y_i+T,\quad y\in\mathbb{R}^d$, which admits the explicit solution $u(r,y)=\sum_{i=1}^d y_i+r$. The problem is solved over the hypercube $[0,0.5]\times[0,0.5]^d$ for dimensions $d=10,20,30,60$. For each experimental setting, we deploy a Physics-Informed Neural Network (PINN) consisting of 5 hidden layers, each containing 50 neurons, and using the hyperbolic tangent (tanh) as the activation function. Training is conducted using the Adam optimizer with a learning rate of 7×10^{-4} and momentum parameters $(0.9,0.99)$, over 10^4 iterations. At every iteration, the network is trained using 2.5×10^3 interior collocation points along with 10^2 boundary points. Evaluation is performed on 1000 interior points and 200 boundary points, uniformly sampled from the same hypercube.

To estimate expectations via the Feynman-Kac formula, we utilize M^n Monte Carlo samples, setting $n=2$, with $M=10$ for the tabulated results and $M=10,\dots,16$ for the inference scaling study. A clipping threshold of $0.5(d+1)$ is applied to both the solution values and gradients in each iteration for both the MLP and SCaSML models. In experiments concerning the linear convection-diffusion scenario (denoted as **LCD** in Table 1), SCaSML achieves a reduction in the relative L^2 error ranging from 20% to 56.9% compared to the baseline surrogate model. Moreover, SCaSML exhibits robust inference scaling, with performance improvements observed as additional inference time is allocated (see Figure 9).

3. SCaSML for Semilinear Parabolic Equations

In this section, we demonstrate that our methodology extends to the semi-linear parabolic equation. Remarkably, we establish that the derived `Law of defect` also takes the form of a semi-linear parabolic equation, enabling the application of Monte Carlo-based algorithms such as the Multilevel Picard Iteration (Hutzenthaler et al., 2019) in the formulation.

3.1. Deriving the `Law of Defect`

To derivate the `Law of defect` that describing the `defect` $\breve{\mathbf{u}}=\mathbf{u}^{\infty}-\hat{\mathbf{u}}$, similar to Section **??**, we define the error of the surrogate model $\hat{u}$ as

$$\epsilon_{\mathrm{PDE}}(r,y):=\frac{\partial}{\partial r}\hat{u}+\langle\mu,\nabla_y\hat{u}\rangle+\frac{1}{2}\mathrm{Tr}(\sigma^*\mathrm{Hess}\;_y\hat{u}\;\sigma)+F(\hat{u},\sigma^*\nabla_y\hat{u}),\ \text{and}\ \hat{g}(y):=\hat{u}(T,y). \tag{6}$$

Then the defect $\breve{u} := u - \hat{u}$ satisfies the following `Law of defect`, derived from (1)–(6) (see Figure 1 b)),

Fact 1. *The $\breve{u}(r, y) := u(r, y) - \hat{u}(r, y)$ is the solution to the following semi-linear Parabolic equation:*

$$\begin{cases} \frac{\partial}{\partial r}\breve{u} + \left\langle \mu, \nabla_y \breve{u} \right\rangle + \frac{1}{2} Tr(\sigma^* Hess\ _y \breve{u}\ \sigma) + \breve{F}(\breve{u}, \sigma^* \nabla_y \breve{u}) = 0, \ on\ [0, T) \times \mathbb{R}^d \\ \breve{u}(T, y) = \breve{g}(y), \ on\ \mathbb{R}^d, \end{cases} \tag{7}$$

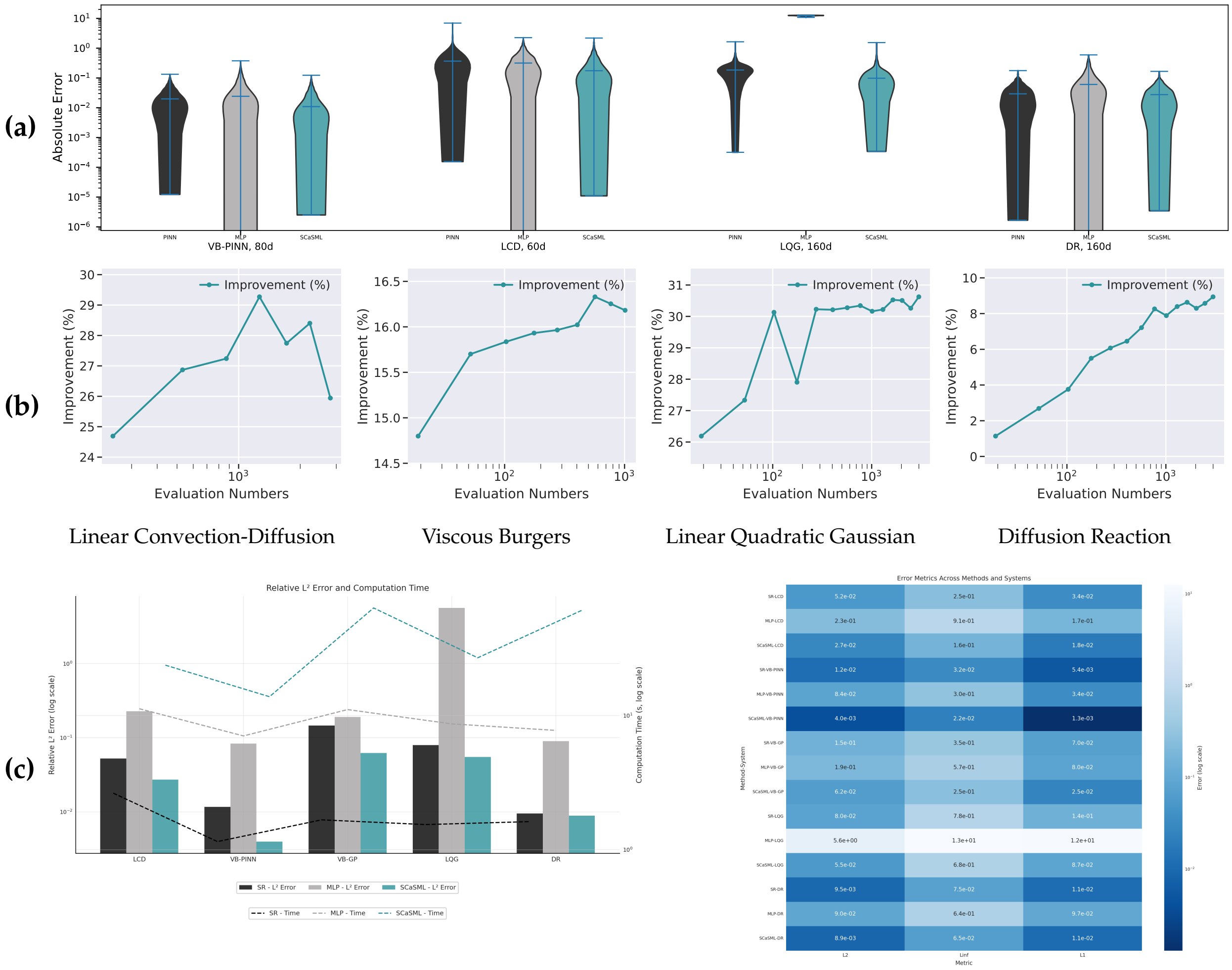
where $\breve{F}(\breve{u}, \sigma^* \nabla_y \breve{u}) = F(\hat{u} + \breve{u}, \sigma^* (\nabla_y \hat{u} + \nabla_y \breve{u})) - F(\hat{u}, \sigma^* \nabla_y \hat{u}) + \varepsilon_{PDE}$, and $\breve{g}(y) = g(y) - \hat{g}(y)$.

Notably, equation (7) is also a semi-linear parabolic equation, owing to the linearity of the Δ operator. Consequently, MLP methods can be employed to solve the `Law of defect`.

3.2. Numerical Experiments

SCaSML consistently delivers the lowest relative L^2 error across all test cases. In particular, for the viscous Burgers equation, relative error reductions range between 16.2% and 66.1%, while for the high-dimensional LQG system the reduction spans 11% to 30.8% compared to the best-performing solver between the surrogate model and the MLP. In addition to achieving superior L^2 performance, SCaSML also attains the lowest L^∞ and L^1 errors in nearly all experiments, underscoring its robustness across various dimensions and norms in large-scale PDE solving tasks. Notably, SCaSML is capable of reducing errors within a factor of approximately 2 to 10 in inference time relative to the surrogate model, facilitated by a parallelized implementation using JAX(Bradbury et al., 2018) and DeepXDE(Lu et al., 2021), which further illustrates its favorable inference scaling properties.

For each PDE, we employed a physics-informed neural network with five hidden layers of 50 neurons per layer and hyperbolic tangent activations. In viscous Burgers, the network was trained using the Adam optimizer with a learning rate of 7×10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.99$ for 10^4 iterations, utilizing 2,500 interior, 100 boundary, and 160 initial sample points uniformly drawn from the domain $0 \leq t \leq 0.5$ and $-0.5 \leq x_i \leq 0.5$ for each spatial dimension. A Gaussian process regression surrogate was trained over 20 iterations via Newton's method, using 1,000 interior and 200 boundary points. For the Hamilton-Jacobi-Bellman equation defined on $0 \leq t \leq 0.5$ and $x \in \mathbb{B}^d$, the network was trained for 2.5×10^3 iterations with 100 interior and 1,000 boundary points per iteration at a learning rate of 10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. The diffusion-reaction system on $0 \leq t \leq 0.5$ and $x \in \mathbb{B}^d$ required 2.5×10^3 iterations with 1,000 interior and 1,000 boundary points under the same optimization parameters. During evaluation, 1,000 interior and 200 boundary points were uniformly sampled from the corresponding geometry. To expedite inference, Hutchinson's method (Girard, 1989; Hutchinson, 1989; Shi et al., 2025) was used to randomly select $d/4$ dimensions for computing Laplacian and divergence operations.



Linear Convection-Diffusion

Viscous Burgers

Linear Quadratic Gaussian

Diffusion Reaction

Figure 2 | **Efficiency and performance of SCaSML methodology.** **a)** Violin Plot for Error distribution of uniformly sampled test points. **b)** Inference Time Scaling of SCaSML. For each equation, we demonstrate how SCaSML improves estimation as the number of inference-time collocation points increases. Our results show that allocating more computational resources at inference consistently leads to more accurate estimations. **c)** Comparative performance of full-history SCaSML against the surrogate model(**SR**, PINN or Gaussian Process) and MLP across multiple PDE systems. Reported metrics include total time (s), relative L^2 error, L^∞ error, and L^1 error.

Viscous Burgers Equation We consider the viscous Burgers equation from (Hutzenthaler et al., 2019):

$$\begin{cases} \dfrac{\partial}{\partial r} u^\infty(r,y) + \left\langle -\left(\dfrac{1}{d} + \dfrac{\sigma_0^2}{2}\right)\mathbf{1}, \nabla_y u^\infty(r,y) \right\rangle + \dfrac{\sigma_0^2}{2}\Delta_y u^\infty(r,y) \\ \qquad + \sigma_0\, u^\infty(r,y) \displaystyle\sum_{i=1}^{d} (\sigma_0 \nabla_y u^\infty(r,y))_i = 0, \quad (r,y) \in [0,T) \times \mathbb{R}^d, \\ u^\infty(T,y) = \dfrac{\exp\left(T + \sum_{i=1}^d y_i\right)}{1 + \exp\left(T + \sum_{i=1}^d y_i\right)}, \quad y \in \mathbb{R}^d, \end{cases} \tag{8}$$

which possesses the explicit solution $u(r,y) = \frac{\exp\left(r+\sum_{i=1}^d y_i\right)}{1+\exp\left(r+\sum_{i=1}^d y_i\right)}$. We solve the problem over the hypercube $[0,0.5] \times [-0.5,0.5]^d$ for dimensions $d = 20,40,60,80$ with $\sigma_0 = \sqrt{2}$. Both the MLP and SCaSML frameworks are applied at level $n = 2$. To mitigate the effect of large outliers that could slow convergence, solutions and gradients are clipped at each level following the thresholding strategy in (Sebastian Becker et al., 2020); specifically, the clipping threshold is set to 1 for the MLP and 0.01 for SCaSML. In the experimental analysis of the viscous Burgers equation, denoted as **VB-PINN** for using the PINN as surrogate model with $M = 10$ and **VB-GP** for using the Gaussian Process as surrogate model with $M = 3$ in Table 1, the SCaSML framework with PINN exhibits a relative L^2 error reduction ranging from 16.2% to 66.1% compared to the standard PINN, while the SCaSML framework with the Gaussian Process shows a reduction between 42.7% and 57.5% compared to the baseline Gaussian Process. Moreover, the SCaSML framework demonstrates advantageous inference scaling properties, characterized by consistently enhanced performance as more inference time is allotted (refer to Figure 10).

Hamilton-Jacobi-Bellman (HJB) Equation The term "curse of dimensionality" was coined by Bellman in dynamic programming (Bellman, 1954). In multi-player optimal control problems, each agent must solve a high-dimensional Hamilton–Jacobi–Bellman (HJB) equation to determine its optimal strategy, which naturally leads to a high-dimensional PDE for the value function. Historically, such high-dimensional PDEs have been largely intractable (Han et al., 2018b). To illustrate the effectiveness of SCaSML debiasing in enhancing neural HJB solvers, we consider a classical linear-quadratic-Gaussian (LQG) control problem.

In a LQG control, the state dynamics are described by $dX_t = 2\sqrt{\Gamma}\, X_t\, dt + \sqrt{2}\, dW_t$ with $X_0 = x$, and the associated cost functional is given by $J\left(\{m_t\}_{t\in[0,T]}\right) = \mathbb{E}\left[\int_0^T \|m_t\|^2\, dt + g(X_T)\right]$, where $\{X_t\}_{t\in[0,T]}$ is the state process, $\{m_t\}_{t\in[0,T]}$ the control process, λ a positive constant quantifying control cost, and $\{W_t\}_{t\in[0,T]}$ is a standard Brownian motion. The objective is to minimize J by choosing an optimal control. The corresponding HJB equation (Yong and Zhou, 1999, Chapter 4) reads

$$\frac{\partial u}{\partial t}(t,x) + \Delta_x u(t,x) - \lambda\, \|\nabla_x u(t,x)\|^2 = 0. \tag{9}$$

The solution $u(t,x)$ evaluated at $t = 0$ represents the optimal cost when starting from state x. By applying Itô's formula, one can verify that the exact solution of (9) with terminal condition $u(T,x) = g(x)$ is given explicitly by $u(t,x) = -\frac{1}{\lambda}\ln\left(\mathbb{E}\left[\exp(-\lambda\, g(X_T))\right]\right)$, where X_T evolves according to the specified dynamics. Following (Hu et al., 2024), we study the case where the terminal condition is given by $u^\infty(T,y) = \log\left(\frac{1+\sum_{i=1}^{d-1}\left[c_{1,i}(y_i - y_{i+1})^2 + c_{2,i} y_{i+1}^2\right]}{2}\right)$, with the coefficients $c_{1,i}$ and $c_{2,i}$ obtained as independent random draws from the uniform distribution on the interval $[0.5, 1.5]$.

To mitigate the impact of large outliers that may slow convergence, we apply a clipping procedure at each level (Sebastian Becker et al., 2020), using a threshold of 10 for the MLP and 0.1 for SCaSML. In experiments with the HJB equation (denoted as **LQG** in Table 1), SCaSML achieves a reduction in the relative L^2 error ranging from 11.7% to 30.8% compared to the baseline surrogate model. Additionally, SCaSML exhibits favorable inference scaling behavior, with performance consistently improving as additional inference time is allocated (see Figure 11).

Diffusion Reaction Equation Given $u^{\star}(r,y) = 1.6 + \sin\left(0.1 \sum_{i=1}^{d} y_i\right) \exp\left(\frac{0.01d(r-1)}{2}\right)$. Following (Han et al., 2018b), we consider a diffusion-reaction equation as in (Gobet and Turkedjiev, 2017):

$$\begin{cases} \frac{\partial}{\partial r} u^{\infty}(r,y) + \frac{1}{2}\Delta_y u^{\infty}(r,y) + \min\{1, (u^{\infty}(r,y) - u^{\star}(r,y))^2\} = 0, & (r,y) \in [0,1) \times \mathbb{R}^d, \\ u^{\infty}(1,y) = 1.6 + \sin\left(0.1 \sum_{i=1}^{d} y_i\right), \quad y \in \mathbb{R}^d, \end{cases} \tag{10}$$

which admits the explicit oscillating solution $u^{\infty}(r,y) = 1.6 + \sin\left(0.1 \sum_{i=1}^{d} y_i\right) \exp\left(\frac{0.01d(r-1)}{2}\right)$. We evaluate the problem for dimensions $d = 100, 120, 140, 160$.

To mitigate the impact of outliers that could impede convergence, we apply clipping at each level, using a threshold of 10 for the MLP and 0.01 for SCaSML. Due to significant fluctuations in the scaling curve, a Hutchinson-type estimator was not employed in this experiment. In our experiments with the diffusion-reaction equation (denoted as **DR** in Table 1), the SCaSML framework achieves a relative L^2 error reduction ranging from 6.6% to 10.9% compared to the baseline surrogate model. Furthermore, SCaSML demonstrates robust inference scaling, with performance consistently improving as additional inference time is allocated (see Figure 12).

4. Provable Improved Convergence Rate of Simulation-Calibrated Scientific Machine Learning

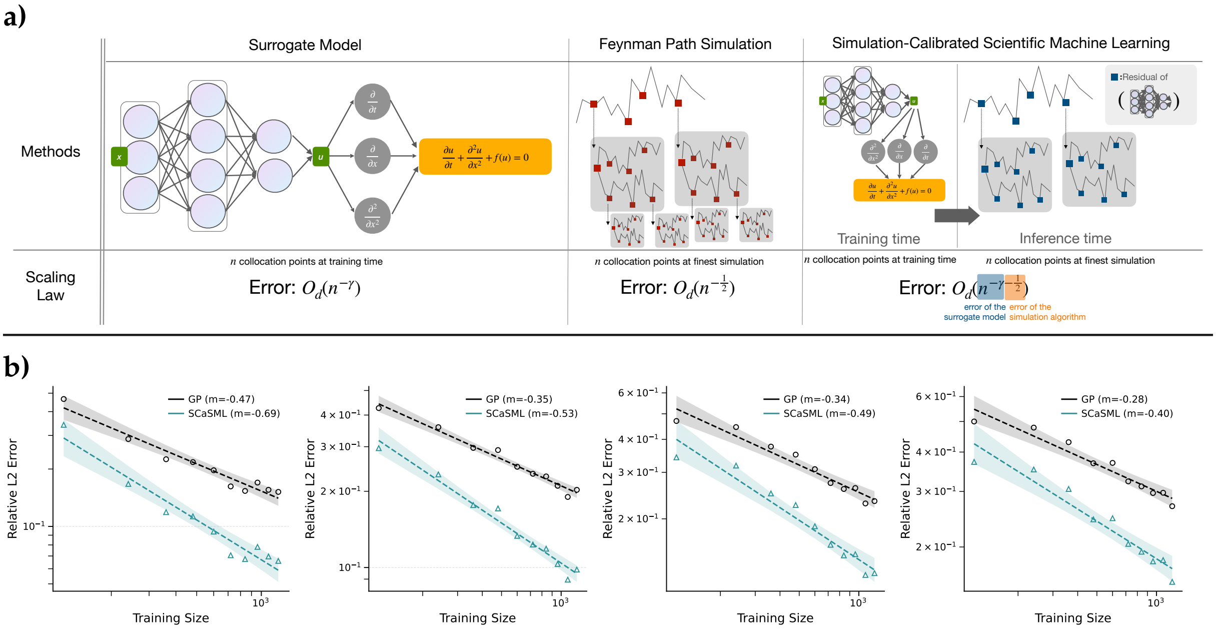
In this section, we present theoretical guarantees of accuracy and show an improved convergence rate for SCaSML methodologies. For simplicity, we assume $\mu = \mathbf{0}$ and $\sigma = s\mathbf{I}_d (s \in \mathbb{R})$ to present the theoretical results and consider the full-history approximation scheme as an example. We can establish a similar theoretical result for SCaSML using quadrature MLP. The detailed theoretical proof can be found in Appendix E.

Assumptions on Surrogate Model Like in language models, where only the powerful enough base models can show an inference time scaling curve, we assume that the error of our learned surrogate model, $\hat{u}$, is limited by an error measure $e(\hat{u})$.

Assumption 1 (Accuracy of the Surrogate Model). *Let* $\sup_{t\in[0,T]} \|\breve{u}^{\infty}(t,\cdot)\|_{W^{1,\infty}} < \infty$. *There exist constants* $C_{F,1}, C_{F,2} > 0$ *such that:*
1. L^{∞} Residual Bound: $\sup_{r\in[0,T], y\in\mathbb{R}^d} |\epsilon_{PDE}(r,y)| \leqslant C_{F,1}\, e(\hat{u})$,

2. $W^{1,\infty}$ Error Bound: $\sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{F,2}\, e(\hat{u})$,

Main Theorems and Proof Idea Our main result establishes that the global L^2 error of the SCaSML estimator is bounded by the product of the surrogate error measure $e(\hat{u})$ and the error



(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

Figure 3 | **Improved Scaling Law for SCaSML. a)** Our SCaSML method stochastically simulates the residual error along the Feynman path, making the simulation difficulty directly proportional to the surrogate model's residual error. This process results in a product defined by the surrogate error measure and the MLP error bound as stated in Theorem 2. Furthermore, by allocating equal amounts of data for both training and inference, SCaSML achieves the improved scaling law detailed in Corollary 4. **b)** By simultaneously increasing the collocation points for both testing and inference, we numerically demonstrate that SCaSML exhibits a faster scaling law than the base surrogate model. Both axes are log scale for all the plots, and the slope γ denotes the polynomial convergence rate. Detailed experiment setting is shown in Appendix G.3.

bound associated with the MLP. Our analysis relies on the fact that the overall L^2 error of MLP is mainly characterized by the Lipschitz continuity of the terminal condition and the magnitude of nonlinearity. Our main observation is that these complexity characterization of the `Law of defect` term can be bounded by the surrogate error, implying that the `Law of defect` is easier to simulate using MLP methods. In Corollary 7, the surrogate error is highlighted in teal and the MLP error in gray. In essence, a highly accurate surrogate (that is, with small $e(\hat{u})$) leads to a proportional reduction in the overall error and, in turn, lower computational complexity to achieve the same precision. Due to page limit, we left the full derivation to Sections E.2.1 and F.2.1.

Theorem 2 (Bound of Global L^2 Error)**.** *Under Assumptions 1, 2, 5 and 6, suppose $p \geqslant 2$, $\alpha \in (\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)})$, $t \in [0,T)$, $x \in \mathbb{R}^d$, $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$. It holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\breve{\mathbf{U}}_{N,M}(t,x) - \breve{\mathbf{u}}(t,x) \right)_\nu \right\|_{L^2} \leq E(M,N) \cdot \Big(C_F \, e(\hat{u}) \Big), \tag{11}$$

where $E(M,N) = \dfrac{\left[e\left(\frac{pN}{2}+1\right)\right]^{\frac{1}{8}} (2C)^{N-1} \exp\left(\beta M^{\frac{1}{2\beta}}\right)}{\sqrt{M^{N-1}}}$. *Here, $\breve{\mathbf{u}}(t,x) = (\breve{u}, \sigma\nabla_x \breve{u})$ is the exact defect we want to estimate and $\breve{\mathbf{U}}_{N,M}(t,x)$ is the estimation using full-history MLP with N levels in total and M^l samples at each level $1 \leqslant l \leqslant N$.*

Remark 3. *Our error bound demonstrates that the full-history SCaSML error satisfies* $Error_{SCaSML} \leqslant \varepsilon(\hat{u}) \cdot \varepsilon_{MLP}$*, where* $\varepsilon(\hat{u})$ *is the error measure for surrogate model* $\hat{u}$ *and* ε_{MLP} *denotes the error bound associated with the naïve full-history MLP method. Consequently, the SCaSML approach preserves the linear dependency on* d *in computational complexity, just as the original MLP framework does. By leveraging a sufficiently accurate base surrogate model, the computational complexity to achieve a global* L^2 *error of* ε *reduces from* $O\big(d\,\varepsilon^{-(2+\delta)}\big)$ *to* $O\big(d\,\varepsilon^{-(2+\delta)}\,e(\hat{u})^{2+\delta}\big)$*, as shown in Corollary 7. For further technical details, please refer to sections E.2.2 and F.2.2.*

Leveraging the established error bound and optimally partitioning the m collocation points between training and inference, we derive in Corollary 4 that the SCaSML method satisfies $O\big(m^{-\gamma-\frac{1}{2}}\,\mathrm{poly}(d)\big)$, compared to the base surrogate model's scaling $e(\hat{u}_m) = O\big(m^{-\gamma}\,\mathrm{poly}(d)\big)$, where $\hat{u}_m$ denotes the surrogate model trained on m collocation points.

Corollary 4 (Provable Improved Scaling Law for SCaSML)**.** *Under Assumptions 1, 2, 5 and 6 , suppose that* $p \geqslant 2$*,* $\alpha \in \left(\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)}\right)$*,* $t \in [0,T)$*,* $x \in \mathbb{R}^d$*, and define* $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$*. Assume that the error at* (t,x) *of the surrogate model decays polynomially with respect to the number of training points; namely,* $e(\hat{u}) = O(m^{-\gamma})$*, for some* $\gamma > 0$*. If we set* $m = (d+1)5^N\,N^{2\beta}$*, for all sufficiently large* m*, the SCaSML procedure improves the error bound from* $O(m^{-\gamma})$ *to* $O\left(m^{-\gamma-\frac{1}{2}+o(1)}\right)$ *when using the same number of collocation points.*

5. Conclusion

In this work, we have demonstrated that integrating simulation-based error correction at inference time can significantly enhance the accuracy of machine-learned PDE solvers. Specifically, the SCaSML framework leverages a novel `Law of defect` to quantify and correct the error of a surrogate model by solving an auxiliary PDE. By simulating error propagation along the Feynman path using techniques based on the Feynman-Kac and Elworthy-Bismut-Li formulas, SCaSML not only reduces the inherent bias of the surrogate but also markedly improves its convergence rate. Numerical experiments confirm that this inference-time correction yields substantial error reductions (ranging from 10% to 80% depending on the method), offering a scalable and theoretically grounded approach to mitigating the curse of dimensionality in high-dimensional PDE problems.

Table 1 | Comparative performance of full-history SCaSML against the surrogate model(**SR**, PINN or Gaussian Process) and MLP across multiple PDE systems. Reported metrics include total time (s), relative L^2 error, L^∞ error, and L^1 error. Bold values indicate the best performance.

		Time (s)			**Relative L^2 Error**			**L^∞ Error**			**L^1 Error**		
		SR	**MLP**	**SCaSML**	**SR**	**MLP**	**SCaSML**	**SR**	**MLP**	**SCaSML**	**SR**	**MLP**	**SCaSML**
LCD	10d	2.64	11.24	23.75	5.24E-02	2.27E-01	**2.73E-02**	2.50E-01	9.06E-01	**1.61E-01**	3.43E-02	1.67E-01	**1.78E-02**
	20d	1.14	7.35	17.59	9.09E-02	2.35E-01	**4.73E-02**	4.52E-01	1.35E+00	**3.28E-01**	9.47E-02	2.37E-01	**4.52E-02**
	30d	1.39	7.52	25.33	2.30E-01	2.38E-01	**1.84E-01**	4.73E+00	1.59E+00	**1.49E+00**	**1.75E-01**	2.84E-01	1.91E-01
	60d	1.13	7.76	35.58	3.07E-01	2.39E-01	**1.32E-01**	3.23E+00	2.05E+00	**1.55E+00**	5.24E-01	4.07E-01	**2.06E-01**
VB-PINN	20d	1.15	7.05	13.82	1.17E-02	8.36E-02	**3.97E-03**	3.16E-02	2.96E-01	**2.16E-02**	5.37E-03	3.39E-02	**1.29E-03**
	40d	1.18	7.49	16.48	3.99E-02	1.04E-01	**2.85E-02**	8.16E-02	3.57E-01	**7.16E-02**	1.97E-02	4.36E-02	**1.21E-02**
	60d	1.19	7.57	19.83	3.97E-02	1.17E-01	**2.90E-02**	8.10E-02	3.93E-01	**7.10E-02**	1.95E-02	4.82E-02	**1.24E-02**
	80d	1.32	7.48	21.99	6.78E-02	1.19E-01	**5.68E-02**	1.89E-01	3.35E-01	**1.79E-01**	3.24E-02	4.73E-02	**2.49E-02**
VB-GP	20d	1.67	11.08	63.38	1.46E-01	1.90E-01	**6.23E-02**	3.54E-01	**5.72E-01**	2.54E-01	7.01E-02	8.00E-02	**2.48E-02**
	40d	1.58	10.91	55.92	1.81E-01	2.20E-01	**8.57E-02**	4.01E-01	8.71E-01	**3.01E-01**	9.19E-02	9.06E-02	**3.82E-02**
	60d	1.59	10.33	56.63	2.40E-01	2.57E-01	**1.28E-01**	3.83E-01	9.50E-01	**2.83E-01**	1.27E-01	9.99E-02	**6.11E-02**
	80d	1.61	10.69	58.53	2.66E-01	3.02E-01	**1.52E-01**	3.61E-01	1.91E+00	**2.61E-01**	1.45E-01	1.09E-01	**7.59E-02**
LQG	100d	1.54	8.67	26.95	7.96E-02	5.63E+00	**5.51E-02**	7.78E-01	1.26E+01	**6.78E-01**	1.40E-01	1.21E+01	**8.68E-02**
	120d	1.25	8.17	27.46	9.37E-02	5.50E+00	**6.64E-02**	9.02E-01	1.27E+01	**8.02E-01**	1.73E-01	1.22E+01	**1.05E-01**
	140d	1.80	8.27	29.72	9.79E-02	5.37E+00	**6.78E-02**	1.00E+00	1.27E+01	**9.00E-01**	1.91E-01	1.23E+01	**1.11E-01**
	160d	1.74	9.07	32.08	1.11E-01	5.27E+00	**9.92E-02**	1.38E+00	1.28E+01	**1.28E+00**	2.15E-01	1.23E+01	**1.79E-01**
DR	100d	1.62	7.75	60.86	9.52E-03	8.99E-02	**8.87E-03**	7.51E-02	6.37E-01	**6.51E-02**	1.13E-02	9.74E-02	**1.11E-02**
	120d	1.26	7.28	65.66	1.11E-02	9.13E-02	**9.90E-03**	7.10E-02	5.74E-01	**6.10E-02**	1.40E-02	9.97E-02	**1.23E-02**
	140d	2.38	7.82	76.90	3.17E-02	8.97E-02	**2.94E-02**	1.79E-01	8.56E-01	**1.69E-01**	3.96E-02	9.77E-02	**3.67E-02**
	160d	1.75	7.42	82.40	3.46E-02	9.00E-02	**3.23E-02**	2.08E-01	8.02E-01	**1.98E-01**	4.32E-02	9.75E-02	**4.02E-02**

References

M. Bachmayr. Low-rank tensor methods for partial differential equations. Acta Numerica, 32: 1–121, 2023.

M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Foundations of Computational Mathematics, 16:1423–1472, 2016.

P. Batlle, Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Error analysis of kernel/gp methods for nonlinear and parametric pdes. arXiv preprint arXiv:2305.04962, 05, 2023.

C. Beck, W. E, and A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. Journal of Nonlinear Science, 29(4):1563–1619, Jan. 2019. ISSN 1432-1467. doi: 10.1007/s00332-018-9525-3. URL `http://dx.doi.org/10.1007/s00332-018-9525-3`.

R. Bellman. Dynamic programming and a new formalism in the calculus of variations. Proceedings of the National Academy of Sciences, 40(4):231–235, 1954. doi: 10.1073/pnas.40.4.231. URL `https://www.pnas.org/doi/abs/10.1073/pnas.40.4.231`.

B. Bouchard, X. Tan, X. Warin, and Y. Zou. Numerical approximation of BSDEs using local polynomial drivers and branching processes. Monte Carlo Methods and Applications, 23(4): 241–263, 2017.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

P. Briand and C. Labart. Simulation of BSDEs by Wiener chaos expansion. The Annals of Applied Probability, 24(3):1129–1171, 2014.

B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024.

S. L. Brunton and J. N. Kutz. Promising directions of machine learning for partial differential equations. Nature Computational Science, 4(7):483–494, 2024.

H.-J. Bungartz and M. Griebel. Sparse grids. Acta numerica, 13:147–269, 2004.

Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart. Solving and learning nonlinear pdes with gaussian processes. Journal of Computational Physics, 447:110668, 2021.

Y. Chen, J. Hoskins, Y. Khoo, and M. Lindsey. Committor functions via tensor networks. Journal of Computational Physics, 472:111646, 2023.

Y. Chen, H. Owhadi, and F. Schäfer. Sparse cholesky factorization for solving nonlinear pdes via gaussian processes, 2024. URL https://arxiv.org/abs/2304.01294.

G. Da Prato and J. Zabczyk. Differentiability of the feynman-kac semigroup and a control application. Atti della Accademia Nazionale dei Lincei. Classe di Scienze Fisiche, Matematiche e Naturali. Rendiconti Lincei. Matematica e Applicazioni, 8(3):183–188, 1997.

W. E and B. Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 6(1):1–12, 2018.

W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse. Multilevel picard iterations for solving smooth semilinear parabolic heat equations. Partial Differential Equations and Applications, 2(6), Nov. 2021. ISSN 2662-2971. doi: 10.1007/s42985-021-00089-5. URL http://dx.doi.org/10.1007/s42985-021-00089-5.

K. D. Elworthy and X.-M. Li. Formulae for the derivatives of heat semigroups. Journal of Functional Analysis, 125(1):252–286, 1994.

K. Gandhi, D. Lee, G. Grand, M. Liu, W. Cheng, A. Sharma, and N. D. Goodman. Stream of search (sos): Learning to search in language, 2024.

C. Geiss and C. Labart. Simulation of BSDEs with jumps by Wiener chaos expansion. Stochastic processes and their applications, 126(7):2123–2162, 2016.

M. B. Giles. Multilevel monte carlo path simulation. Operations research, 56(3):607–617, 2008.

M. B. Giles. Multilevel monte carlo methods. Acta numerica, 24:259–328, 2015.

A. Girard. A fast 'monte-carlo cross-validation'procedure for large least squares problems with noisy data. Numerische Mathematik, 56:1–23, 1989.

E. Gobet and P. Turkedjiev. Adaptive importance sampling in least-squares monte carlo algorithms for backward stochastic differential equations. Stochastic Processes and their Applications, 127(4):1171–1203, 2017. ISSN 0304-4149. doi: https://doi.org/10.1016/j.spa.2016.07.011. URL https://www.sciencedirect.com/science/article/pii/S0304414916301235.

P. Grohs, A. Jentzen, and D. Salimova. Deep neural network approximations for solutions of pdes based on monte carlo algorithms, June 2022. ISSN 2662-2971. URL http://dx.doi.org/10.1007/s42985-021-00100-z.

J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences, 115(34):8505–8510, 2018a.

J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences, 115(34):8505–8510, 2018b. doi: 10.1073/pnas.1718942115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1718942115.

P. Henry-Labordère, N. Oudjane, X. Tan, N. Touzi, and X. Warin. Branching diffusion representation of semilinear PDEs and Monte Carlo approximation. Annales de l'Institut Henri Poincaré, Probabilités et Statistiques, 55(1):184–210, 2019.

J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural pde solvers with convergence guarantees. arXiv preprint arXiv:1906.01200, 06, 2019.

Z. Hu, K. Shukla, G. E. Karniadakis, and K. Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. Neural Networks, 176:106369, Aug. 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2024.106369. URL http://dx.doi.org/10.1016/j.neunet.2024.106369.

M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. Communications in Statistics-Simulation and Computation, 18(3):1059–1076, 1989.

M. Hutzenthaler and T. Kruse. Multilevel picard approximations of high-dimensional semilinear parabolic differential equations with gradient-dependent nonlinearities. SIAM Journal on Numerical Analysis, 58(2):929–961, Jan. 2020. ISSN 1095-7170. doi: 10.1137/17m1157015. URL http://dx.doi.org/10.1137/17M1157015.

M. Hutzenthaler, A. Jentzen, T. Kruse, et al. On multilevel picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. Journal of Scientific Computing, 79(3): 1534–1571, 2019.

M. Hutzenthaler, A. Jentzen, T. Kruse, T. Anh Nguyen, and P. von Wurstemberger. Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476(2244), Dec. 2020a. ISSN 1471-2946. doi: 10.1098/rspa.2019.0630. URL http://dx.doi.org/10.1098/rspa.2019.0630.

M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen. Multilevel picard approximations for high-dimensional semilinear second-order pdes with lipschitz nonlinearities, 2020b.

M. Hutzenthaler, A. Jentzen, and T. Kruse. Overcoming the curse of dimensionality in the numerical approximation of parabolic partial differential equations with gradient-dependent nonlinearities. Foundations of Computational Mathematics, 22(4):905–966, July 2021. ISSN 1615-3383. doi: 10.1007/s10208-021-09514-y. URL http://dx.doi.org/10.1007/s10208-021-09514-y.

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. Nature Reviews Physics, 3(6):422–440, 2021.

Z. Long, Y. Lu, X. Ma, and B. Dong. Pde-net: Learning pdes from data. In International conference on machine learning, pages 3208–3216. PMLR, 2018.

Z. Long, Y. Lu, and B. Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. Journal of Computational Physics, 399:108925, 2019.

L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. SIAM Review, 63(1):208–228, Jan. 2021. ISSN 1095-7200. doi: 10.1137/19m1274067. URL http://dx.doi.org/10.1137/19M1274067.

Y. Lu, J. Blanchet, and L. Ying. Sobolev acceleration and statistical optimality for learning elliptic equations via gradient descent. Advances in Neural Information Processing Systems, 35:33233–33247, 2022a.

Y. Lu, H. Chen, J. Lu, L. Ying, and J. Blanchet. Machine learning for elliptic PDEs: Fast rate generalization bound, neural scaling law and minimax optimality. In International Conference on Learning Representations, 2022b. URL https://openreview.net/forum?id=mhYUBYNoGz.

R. Nickl, S. van de Geer, and S. Wang. Convergence rates for penalized least squares estimators in pde constrained regression problems. SIAM/ASA Journal on Uncertainty Quantification, 8(1):374–413, 2020.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017. URL https://arxiv.org/abs/1711.10561.

Research Group on Stochastic Analysis, University of Duisburg-Essen. Multilevel picard approximation, 2025. URL https://www.uni-due.de/mathematik/ag_stochastische_analysis/mlp. Accessed: March 26, 2025.

L. Richter, L. Sallandt, and N. Nüsken. Solving high-dimensional parabolic pdes using the tensor train format. In International Conference on Machine Learning, pages 8998–9009. PMLR, 2021.

S. B. Sebastian Becker, R. B. Ramon Braunwarth, M. H. Martin Hutzenthaler, A. J. Arnulf Jentzen, and P. v. W. Philippe von Wurstemberger. Numerical simulations for full history recursive multilevel picard approximations for systems of high-dimensional partial differential equations. Communications in Computational Physics, 28(5):2109–2138, Jan. 2020. ISSN 1815-2406. doi: 10.4208/cicp.oa-2020-0130. URL http://dx.doi.org/10.4208/cicp.OA-2020-0130.

Z. Shi, Z. Hu, M. Lin, and K. Kawaguchi. Stochastic taylor derivative estimator: Efficient amortization for arbitrary differential operators, 2025. URL https://arxiv.org/abs/2412.00088.

J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. Journal of computational physics, 375:1339–1364, 2018.

C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024.

W. Suo and W. Zhang. A novel paradigm for solving pdes: multi scale neural computing. arXiv preprint arXiv:2312.06949, 12, 2023.

M. Unser. A unifying representer theorem for inverse problems and machine learning. Foundations of Computational Mathematics, 21(4):941–960, 2021.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models, 2022.

E. Weinan, J. Han, and A. Jentzen. Algorithms for solving high dimensional pdes: from nonlinear monte carlo to machine learning. Nonlinearity, 35(1):278, 2021.

Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang. An empirical analysis of compute-optimal inference for problem-solving with language models, 2024.

S. Yang, S. W. Wong, and S. Kou. Inference of dynamic systems from noisy and sparse data via manifold-constrained gaussian processes. Proceedings of the National Academy of Sciences, 118(15):e2020397118, 2021.

J. Yong and X. Y. Zhou. Stochastic controls: Hamiltonian systems and hjb equations. 1999. URL `https://api.semanticscholar.org/CorpusID:118042879`.

J. Yu, L. Lu, X. Meng, and G. E. Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. Computer Methods in Applied Mechanics and Engineering, 393:114823, Apr. 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2022.114823. URL `http://dx.doi.org/10.1016/j.cma.2022.114823`.

E. Zhang, A. Kahana, E. Turkel, R. Ranade, J. Pathak, and G. E. Karniadakis. A hybrid iterative numerical transferable solver (hints) for pdes based on deep operator network and relaxation methods. arXiv preprint arXiv:2208.13273, 8, 2022.

M. Zhou, J. Han, M. Rachh, and C. Borges. A neural network warm-start approach for the inverse acoustic obstacle scattering problem. Journal of Computational Physics, 490:112341, 2023.

A. Outline of the Appendix

Appendix A. **Preliminaries.** This section introduces two essential components of our experiments: the surrogate models in SCaSML and the Multilevel Picard (MLP) iterations.

1. Surrogate Models for PDEs. We present the preliminary on the surrogate models we used for solving partial differential equations.
   (a) **Physics-Informed Neural Network (PINN).** Details on training physics-informed neural networks as base surrogate models.
   (b) **Gaussian Processes.** (Chen et al., 2021; Yang et al., 2021) approximate the solution to a given PDE by computing the maximum a posteriori (MAP) estimator of a Gaussian process that is conditioned to satisfy the PDE at a limited number of collocation points. While this formulation naturally leads to an infinite-dimensional optimization problem, it can be transformed into a finite-dimensional one by introducing auxiliary variables that represent the values of the solution's derivatives at these collocation points—a strategy that extends the representer theorem familiar from Gaussian process regression. The resulting finite-dimensional problem features a quadratic objective function subject to nonlinear constraints and is solved using a variant of the Gauss–Newton method.
2. Quadrature and full-history Multilevel Picard Iterations. In this section, we provide an overview of the Mutlilevel Picard Iteration (MLP) methods for solving semi-linear Parabolic equation is provided, including two variants:quadrature MLP(E et al., 2021) and full-history MLP(Hutzenthaler et al., 2021).

Appendix C. **Algorithm.** In this section, we present the full procedure of SCaSML algorithms.

Appendix D. **Proof Settings.** This section details the common assumptions and notation for all MLP methods, and presents regularity conditions on surrogate models.

1. Notation. We adopt the notation system introduced in (Hutzenthaler et al., 2021).
2. Regularity Assumptions for Surrogate Model. We introduce regularity conditions on surrogate models.

Appendix E. **Proof for Quadrature Multilevel Picard Iteration.** In this section, we establish the convergence rate improvement for SCaSML using quadrature integration.

1. Settings. SCaSML needs a good base surrogate model. In this section, we specify the accuracy assumptions for the base surrogate model.
2. Main Results. The primary theoretical results include:
   (a) **Global L^2 Error Bound.** The error analysis employs a factor replacement strategy, with the flexibility of choosing the inference time computation cost M.
   (b) **Computational Complexity Bound.** By setting optimal relationship between training and inference time computational cost, we show that SCaSML can reduce the computational complexity from $O(d\varepsilon^{-(4+\delta)})$ to $O(de(\hat{u})^{4+\delta}\varepsilon^{-(4+\delta)})$.

Appendix F. **Proof for full-history Multilevel Picard Iteration.** This section mirrors the previous one but focuses on the full-history variant.

1. Settings. SCaSML needs a good base surrogate model. In this section, we specify the accuracy assumptions for the base surrogate model.
2. Main Results. The key theoretical findings are:
   (a) **Global L^2 Error Bound.** The error analysis employs a factor replacement strategy, with the flexibility of choosing the inference time computation cost M.

(b) **Computational Complexity Bound.** By setting optimal relationship between training and inference time computational cost, we show that SCaSML can reduce the computational complexity from $O(d\varepsilon^{-(2+\delta)})$ (MLP) to $O(de(\hat{u})^{2+\delta}\varepsilon^{-(2+\delta)})$. We also show a improved scaling law for SCaSML.

Appendix G. **Auxiliary Experimental Results.** This section compiles supplementary experimental data that support our theoretical claims, including violin plots of the absolute error distributions and inference time scaling law curves illustrating the improved convergence rate.

B. Preliminary

B.1. Surrogate Models for PDEs

In our experiments, we employ two surrogate models to solve high-dimensional PDEs:a Physics-Informed Neural Network (PINN) and a Gaussian Process (GP) regression model. Both models are implemented in `JAX` (Bradbury et al., 2018) and `DeepXDE` (Lu et al., 2021) to leverage efficient parallelization and runtime performance. Furthermore, Hutchinson's estimator technique 3 as delineated in (Shi et al., 2025) is incorporated during the training process to substantially decrease GPU memory consumption, applicable to both the training and inference stages of Physics-Informed Neural Networks (PINN), as well as the inference phase of Gaussian Processes.

B.1.1. Physics-Informed Neural Network (PINN)

Physics-Informed Neural Networks (PINNs) are designed to approximate solutions of PDEs by embedding physical laws into the learning process. In our framework, the neural network $\hat{u}(t,x)$ with parameters θ approximates the true solution $u^{\infty}(t,x)$ of the given PDE. The training loss is constructed as a weighted sum of several components, each designed to enforce key aspects of the problem's constraints.

The first component is the PDE loss, which ensures that the network output adheres to the governing differential equation. This is achieved by penalizing deviations from the expected behavior defined by the differential operator, evaluated at a set of interior collocation points $\{(t_k,x_k)\}_{k=1}^{S_1}$. The PDE loss is defined as

$$\mathcal{L}_{\mathrm{PDE}}(\theta)=\frac{1}{S_1}\sum_{k=1}^{S_1}\left|\frac{\partial\hat{u}}{\partial r}(t_k,x_k)+\frac{\sigma^2}{2}\Delta_y\hat{u}(t_k,x_k)+F\big(\hat{u},\sigma\nabla_y\hat{u}\big)(t_k,x_k)\right|^2. \tag{12}$$

In order to satisfy the prescribed boundary conditions, the model employs a Dirichlet boundary loss. This term minimizes the difference between the network output and the given boundary values $h(x_k)$ at selected boundary points $\{(t_k,x_k)\}_{k=1}^{S_2}$, and is expressed as

$$\mathcal{L}_{\mathrm{Dir}}(\theta)=\frac{1}{S_2}\sum_{k=1}^{S_2}|\hat{u}(t_k,x_k)-h(x_k)|^2. \tag{13}$$

Moreover, the initial conditions of the problem are enforced by an initial loss component. This ensures that the solution at time $t=0$ matches the known initial data $q(x_k)$ for the points $\{(0,x_k)\}_{k=1}^{S_3}$:

$$\mathcal{L}_{\mathrm{initial}}(\theta)=\frac{1}{S_3}\sum_{k=1}^{S_3}|\hat{u}(0,x_k)-q(x_k)|^2. \tag{14}$$

The overall training objective is then formulated as a combination of these losses, with each term scaled by its corresponding weighting coefficient:

$$\mathcal{L}(\theta) = \alpha_1 \, \mathcal{L}_{\text{PDE}}(\theta) + \alpha_2 \, \mathcal{L}_{\text{Dir}}(\theta) + \alpha_3 \, \mathcal{L}_{\text{initial}}(\theta). \tag{15}$$

This formulation ensures that the PINN not only fits the observed data but also rigorously respects the underlying physical laws, boundary conditions, and initial conditions governing the PDE.

B.1.2. Gaussian Processes

In this section, we review the Gaussian Process (GP) framework developed in (Chen et al., 2021, 2024; Yang et al., 2021) to solve nonlinear PDEs. Consider solving a semi-linear parabolic PDE

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x) = \tau\big(u(t,x),\, \Delta_x u(t,x),\, \operatorname{div}_x u(t,x)\big), & \forall (t,x) \in [0,T] \times \mathbb{R}^d, \\ u(T,x) = g(x), & \forall x \in \mathbb{R}^d, \end{cases} \tag{16}$$

where τ is a nonlinear function of the solution and its derivatives, and g specifies the terminal condition.

The GP Framework Consider one already sample M_{in} interior points and M_{bd} boundary points, denoted as $\mathbf{x}_{\text{in}} = \{\mathbf{x}_{\text{in}}^1, ..., \mathbf{x}_{\text{in}}^{M_{\text{in}}}\} \subset [0,T] \times \mathbb{R}^d$ and $\mathbf{x}_{\text{bd}} = \{\mathbf{x}_{\text{bd}}^1, ..., \mathbf{x}_{\text{bd}}^{M_{\text{bd}}}\} \subset \{T\} \times \mathbb{R}^d$. Then, we assign an unknown GP prior to the unknown function u with mean 0 and covariance function $K : ([0,T] \times \mathbb{R}^d) \times ([0,T] \times \mathbb{R}^d) \to \mathbb{R}$, the method aims to compute the maximum a posterior estimator of the GP given the sampled PDE data, which leads to the following optimization problem

$$\begin{cases} \underset{u \in \mathcal{U}}{\text{minimize}} & \|u\| \\ \text{s.t.} & \frac{\partial u}{\partial t}(\mathbf{x}_{\text{in}}^m) = \tau(u(\mathbf{x}_{\text{in}}^m), \Delta u(\mathbf{x}_{\text{in}}^m), \operatorname{div}_x u(\mathbf{x}_{\text{in}}^m)), \quad \text{for} \quad m = 1, \dots, M_{\text{in}}, \\ & u(\mathbf{x}_{\text{bd}}^m) = g(\mathbf{x}_{\text{bd}}^m), \quad \text{for} \quad m = 1, \dots, M_{\text{bd}}. \end{cases} \tag{17}$$

Here, $\|\cdot\|$ is the Reproducing Kernel Hilbert Space(RKHS) norm corresponding to the kernel/covariance function K. Regarding consistency, once K is sufficiently regular, the above solution will converge to the exact solution of the PDE when $M_{\text{in}}, M_{\text{bd}} \to \infty$; see (Batlle et al., 2023, Theorem 1.2).

We denote the measurement functions by

$$\begin{aligned} &\varphi_m^1(u) : u \to \delta_{\mathbf{x}_{\text{in}}^m} \circ u, 1 \leqslant m \leqslant M_{\text{in}}, && \varphi_m^2(u) : u \to \delta_{\mathbf{x}_{\text{bd}}^m} \circ u, 1 \leqslant m \leqslant M_{\text{bd}}, \\ &\varphi_m^3(u) : u \to \delta_{\mathbf{x}_{\text{in}}^m} \circ \Delta_x u, 1 \leqslant m \leqslant M_{\text{in}}, && \varphi_m^4(u) : u \to \delta_{\mathbf{x}_{\text{in}}^m} \circ \frac{\partial u}{\partial t}, 1 \leqslant m \leqslant M_{\text{in}}, \\ &\varphi_m^5(u) : u \to \delta_{\mathbf{x}_{\text{in}}^m} \circ \operatorname{div}_x u, 1 \leqslant m \leqslant M_{\text{in}}, \end{aligned} \tag{18}$$

where $\delta_{\mathbf{x}}$ is the Dirac delta function centered at $\mathbf{x}$. These functions belong to $\mathcal{U}^*$, the dual space of $\mathcal{U}$, for sufficiently regular kernel functions. We further use the shorthand notation $\varphi^1, \varphi^3, \varphi^4, \varphi^5$ for M_{in} dimensional vectors and φ^2 for M_{bd} dimensional vectors as finite dimensional representation for corresponding features. We use $[\cdot,\cdot]$ to denote the primal-dual pairing, such that for $u \in \mathcal{U}$ and $\varphi_m^i \in \mathcal{U}^*, \forall i$ it holds that $[u, \varphi_m^i] = \int u(\mathbf{x}) \varphi_m^i(\mathbf{x}) d\mathbf{x}$. For instance, for φ_m^3 we have

$[u, \varphi^3_m] = \int u(\mathbf{x})\varphi^3_m(\mathbf{x})d\mathbf{x} = \frac{\partial u}{\partial t}(\mathbf{x}_m)$. Based on the defined notation, we can rewrite the MAP problem (17) as

$$\begin{cases} \underset{u\in\mathcal{U}}{\text{minimize}} & \|u\| \\ \text{s.t.} & z^{(1)}_m = \varphi^{(1)}_m(u), z^{(3)}_m = \varphi^{(3)}_m(u), z^{(4)}_m = \varphi^{(4)}_m(u), z^{(5)}_m = \varphi^{(5)}_m(u), \quad m = 1, \ldots, M_{\text{in}}, \\ & z^{(1)}_m = \varphi^{(1)}_m(u), m = 1, \ldots, M_{\text{bd}}, \\ & z^{(4)}_m = \tau(z^{(1)}_m, z^{(3)}_m, z^{(5)}_m), \quad m = 1, \ldots, M_{\text{in}}, \\ & z^{(2)}_m = g(\mathbf{x}^m_{bd}), \quad m = 1, \ldots, M_{\text{bd}}. \end{cases} \tag{19}$$

Finite Dimensioanl Representation via Representer Theorem According to Representer Theorem (Chen et al., 2021; Unser, 2021) show that although the original MAP problem (17) is an infinite-dimensional optimization problem, the minimizer enjoys a finite-dimensional structure

$$u^\dagger(\mathbf{x}) = K(\mathbf{x}, \varphi)\alpha \tag{20}$$

where $K(\mathbf{x}, \varphi)$ is the $(4M_{\text{in}} + M_{\text{bd}})$ dimensional vector with entries $\int K(\mathbf{x}, \mathbf{x}')\varphi_j(\mathbf{x}')d\mathbf{x}'$ (here the integral notation shall be interpreted as the primal-dual pairing as above), i.e.

$$K(\mathbf{x}, \varphi) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_{\text{in}}) & K(\mathbf{x}, \mathbf{x}_{\text{bd}}) & \Delta_{x'}K(\mathbf{x}, \mathbf{x}_{\text{in}}) & \frac{\partial}{\partial t}K(\mathbf{x}, \mathbf{x}_{\text{in}}) & \operatorname{div}_{x'} K(\mathbf{x}, \mathbf{x}_{\text{in}}) \end{bmatrix} \in \mathbb{R}^{1\times(4M_{\text{in}}+M_{\text{bd}})}, \tag{21}$$

and $\alpha \in \mathbb{R}^{4M_{\text{in}}+M_{\text{bd}}}$ is the unknown coeficients. Based on the finite dimensional representation (20), we know

$$\left[z^{(1)\top}, z^{(2)\top}, z^{(3)\top}, z^{(4)\top}, z^{(5)\top}\right]^\top = K(\varphi, \varphi)\alpha, \tag{22}$$

wher $z^{(1)} = [\varphi^1_1(u), \varphi^1_2(u), \cdots, \varphi^1_{M_{\text{in}}}(u)]^\top \in \mathbb{R}^{M_{\text{in}}}, z^{(2)} = [\varphi^2_1(u), \varphi^2_2(u), \cdots, \varphi^2_{M_{\text{bd}}}(u)]^\top \in \mathbb{R}^{M_{\text{bd}}}, z^{(3)} = [\varphi^3_1(u), \varphi^3_2(u), \cdots, \varphi^3_{M_{\text{in}}}(u)]^\top \in \mathbb{R}^{M_{\text{in}}}, z^{(4)} = [\varphi^4_1(u), \varphi^4_2(u), \cdots, \varphi^4_{M_{\text{in}}}(u)]^\top \in \mathbb{R}^{M_{\text{in}}}, z^{(5)} = [\varphi^5_1(u), \varphi^5_2(u), \cdots, \varphi^5_{M_{\text{in}}}(u$ $\mathbb{R}^{M_{\text{in}}}$, and $K(\varphi, \varphi)$ is the kernel matrix as the $(4M_{\text{in}} + M_{\text{bd}}) \times (4M_{\text{in}} + M_{\text{bd}})$ matrix with entries $\int K(\mathbf{x}, \mathbf{x}')\varphi_m(\mathbf{x})\varphi_j(\mathbf{x}')d\mathbf{x}d\mathbf{x}'$ where φ_m denotes the entries of φ. Precisely $K(\varphi, \varphi)$ can be written down explicitly as:

$$K(\varphi, \varphi) = \begin{bmatrix} K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{bd}}) & \Delta_{x'}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \operatorname{div}_{x'} K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) \\ K(\mathbf{x}_{\text{bd}}, \mathbf{x}'_{\text{in}}) & K(\mathbf{x}_{\text{bd}}, \mathbf{x}'_{\text{bd}}) & \Delta_{x'}K(\mathbf{x}_{\text{bd}}, \mathbf{x}'_{\text{in}}) & \frac{\partial}{\partial t}K(\mathbf{x}_{\text{bd}}, \mathbf{x}'_{\text{bd}}) & \operatorname{div}_{x'} K(\mathbf{x}_{\text{bd}}, \mathbf{x}'_{\text{in}}) \\ \Delta_x K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \Delta_x K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{bd}}) & \Delta_x \Delta_{x'}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \Delta_x \frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \Delta_x \operatorname{div}_{x'} K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) \\ \frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{bd}}) & \frac{\partial}{\partial t}\Delta_{x'}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \frac{\partial}{\partial t}\frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \frac{\partial}{\partial t} \operatorname{div}_{x'} K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) \\ \operatorname{div}_x K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \operatorname{div}_x K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{bd}}) & \operatorname{div}_x \Delta_{x'}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \operatorname{div}_x \frac{\partial}{\partial t}K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) & \operatorname{div}_x \operatorname{div}_{x'} K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) \end{bmatrix}, \tag{23}$$

Here we adopt the convention that if the variable inside a function is a set, it means that this function is applied to every element in this set; the output will be a vector or a matrix, e.g. $K(\mathbf{x}_{\text{in}}, \mathbf{x}'_{\text{in}}) = \exp\left(-\frac{\|\mathbf{x}^m_{\text{in}} - \mathbf{x}^j_{\text{in}}\|^2_2}{2(\sigma\sqrt{d})^2}\right), 1 \leqslant m, j \leqslant M_{\text{in}}, \in \mathbb{R}^{M_{\text{in}}\times M_{\text{in}}}$ in the Gaussian kernel of our numerical experiment, where σ is the variance of the equation. Thus the finite dimensional representation (20) can be rewritten in terms of the function (derative) values

$$u^\dagger(\mathbf{x}) = K(\mathbf{x}, \varphi)K(\varphi, \varphi)^{-1}z^\dagger, \tag{24}$$

where $z^\dagger = \left[z^{(1)\top}, z^{(2)\top}, z^{(3)\top}, z^{(4)\top}, z^{(5)\top}\right]^\top \in \mathbb{R}^{4M_{\text{in}}+M_{\text{bd}}}$.

Plug the finite-dimensional representation (24) to the original MAP problem (19) we have that $z^\dagger$ is the solution to the following finite-dimensional quadratic optimization optimization problem with nonlinear constraints

$$\begin{aligned} &\min_{z \in \mathbb{R}^{4M_{\text{in}}+M_{\text{bd}}}} \quad z^\top K(\varphi,\varphi)^{-1} z \\ &\text{subject to} \\ &\quad z_m^{(4)} = \tau(z_m^{(1)}, z_m^{(3)}, z_m^{(5)}), \quad m = 1,\ldots,M_{\text{in}}, \\ &\quad z_m^{(2)} = g(\mathbf{x}_{bd}^m), \quad m = 1,\ldots,M_{\text{bd}}. \end{aligned} \tag{25}$$

Solving the Optimization Formulation To develop efficient optimization algorithms for (25), observing that the constraints $z_m^{(4)} = \tau\big(z_m^{(1)}, z_m^{(3)}, z_m^{(5)}\big)$ and $z_m^{(2)} = g\big(\mathbf{x}_{bd}^m\big)$ express $z_m^{(4)}$ and $z_m^{(2)}$ in terms of the other variables, (Chen et al., 2021, 2024) reformulate the optimization problem as an unconstrained problem

$$\min_{z^{(1)},\, z^{(3)},\, z^{(5)} \in \mathbb{R}^{M_{\text{in}}}} [z^{(1)}; g(\mathbf{x}_{bd}); z^{(3)}; \tau(z^{(1)}, z^{(3)}, z^{(5)}); z^{(5)}]^\top K(\varphi,\varphi)^{-1} [z^{(1)}; g(\mathbf{x}_{bd}); z^{(3)}; \tau(z^{(1)}, z^{(3)}, z^{(5)}); z^{(5)}].$$

We apply Sparse Cholesky decomposition to the positive-definite $(K(\varphi,\varphi) + \eta I)$ as LL^T. In turn, $b^T(K(\varphi,\varphi) + \eta I)^{-1} b = b^T(LL^T)^{-1} b = (L^{-1}b)^T(L^{-1}b) = \|L^{-1}b\|_2^2$. Hence, the loss function is defined as $\mathcal{J}(z^{(1)}, z^{(3)}, z^{(5)}) = \|L^{-1}b\|^2$. Optimization is carried out via a Newton method in 20 iterations. We initialize $z^{(1)},\ z^{(3)},\ z^{(5)} \in \mathbb{R}^{M_{\text{in}}}$ following $N(0, 10^{-6} I_{M_{\text{in}}})$. In each iteration, the gradient $\nabla\mathcal{J}$ and Hessian $\nabla^2\mathcal{J}$ are computed via automatic differentiation, and the Newton direction Δz is obtained by solving $\big(\nabla^2\mathcal{J} + \lambda I\big)\,\Delta z = -\nabla\mathcal{J}$, where $\lambda = 10^{-4}$ is an regularization parameter. Then, update $\mathcal{J}$ at Newton direction with step size $\alpha = 1$. Early stopping is triggered when the gradient norm falls below 10^{-5}. Finally, to apply the representer theorem in 24, the algorithm solves the linear system $(K(\varphi,\varphi) + \eta I) w^\dagger = z^\dagger$ to obtain the weight vector $w^\dagger$ and the final PDE solution is given as $u^\dagger(\mathbf{x}) = K(\mathbf{x},\varphi) w^\dagger$.

B.2. Quadrature Multilevel Picard Iterations and full-history Multilevel Picard Iterations

Multilevel Picard Iteration (MLP) method (Hutzenthaler et al., 2019) is a simulation-based solver which solves a semilinear parabolic PDEs (Han et al., 2018a; Hutzenthaler et al., 2019; Weinan et al., 2021), represented as the following.

$$\begin{cases} \dfrac{\partial}{\partial r} u^\infty + \big\langle \mu, \nabla_y u^\infty \big\rangle + \dfrac{1}{2}\mathrm{Tr}(\sigma^* \mathrm{Hess}\, u^\infty\, \sigma) + F(u^\infty, \sigma^* \nabla_y u^\infty) = 0, \text{ on } [0,T) \times \mathbb{R}^d \\ u^\infty(T, y) = g(y), \text{ on } \mathbb{R}^d. \end{cases} \tag{26}$$

where $T > 0, d \in \mathbb{N}, g : \mathbb{R}^d \to \mathbb{R}, u^\infty : [0,T] \times \mathbb{R}^{d+1} \to \mathbb{R}, \mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$. Additionally, let σ be a regular function mapping $[0,T] \times \mathbb{R}^d$ to a real $d \times d$ invertible matrix.

The MLP method reformulates the PDE into a fixed-point problem using the Feynman–Kac formula to represent the solution as the expected value of a stochastic process's functional. A Picard scheme iteratively solves this fixed-point problem. The MLP method employs a multilevel Monte Carlo approach(Giles, 2008), blending coarse and fine discretizations and allocating more samples to deeper iterations to control variance. This strategy ensures computational costs increase moderately with accuracy. According to Feynman–Kac and Bismut-Elworthy-Li formula(Da Prato and Zabczyk, 1997; Elworthy and Li, 1994), the solution $\mathbf{u}^\infty = (u, \sigma^* \nabla_y u)$ of semilinear parabolic PDE (26) satisfies the fixed-point equation $\Phi(\mathbf{u}^\infty) = \mathbf{u}^\infty$ where Φ:Lip$([0,T] \times \mathbb{R}^d, \mathbb{R}^{1+d}) \to$ Lip$([0,T] \times \mathbb{R}^d, \mathbb{R}^{1+d})$ is defined as

$$\begin{aligned}(\Phi(\mathbf{v}))\,(s,x) =& \mathbb{E}\left[g(X_T^{s,x})\left(1, \frac{[\sigma(s,x)]^*}{T-s}\int_s^T \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right] \\ &+ \int_s^T \mathbb{E}\left[F(\mathbf{v}(t,X_t^{s,x}))\left(1, \frac{[\sigma(s,x)]^*}{t-s}\int_s^t \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right] dt.\end{aligned} \tag{27}$$

Here $X_t^{s,x}$ and $D_t^{s,x}$ are defined as

$$\begin{aligned}X_t^{s,x} &= x + \int_s^t \mu(r,X_r^{s,x})dr + \sum_{j=1}^d \int_s^t \sigma_j(r,X_r^{s,x})dW_r^j, \\ D_t^{s,x} &= \mathrm{I}_{\mathbb{R}^{d\times d}} + \int_s^t (\frac{\partial}{\partial x}\mu)(r,X_r^{s,x})D_r^{s,x}dr + \sum_{j=1}^d \int_s^t (\frac{\partial}{\partial x}\sigma_j)(r,X_r^{s,x})D_r^{s,x}dW_r^j.\end{aligned} \tag{28}$$

where $W_t : [0,T] \times \Omega \to \mathbb{R}^d$ is a standard $(\mathbb{F}_t)_{t\in[0,T]}$-adapted Brownian motion.

The Feynman-Kac formula gives

$$u^\infty(s,x) = \mathbb{E}[g(X_T^{s,x})] + \int_s^T \mathbb{E}[F(u^\infty(t,X_t^{s,x}), [\sigma(t,X_t^{s,x})]^*(\nabla_y u^\infty)(t,X_t^{s,x}))]dt. \tag{29}$$

Note that $\sigma^*\nabla_y u^\infty$ appeared on the right-hand side in the fixed point iteration, which necessitates a new representation formula of it to be simultaneous with 29. And that is Bismut-Elworthy-Li formula(Da Prato and Zabczyk, 1997; Elworthy and Li, 1994), which gives

$$\begin{aligned}[\sigma(s,x)]^*(\nabla_y u^\infty)(s,x) =& \mathbb{E}\left[g(X_T^{s,x})\frac{[\sigma(s,x)]^*}{T-s}\int_s^T \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right] \\ &+ \int_s^T \mathbb{E}\Bigg[F(u^\infty(t,X_t^{s,x}), [\sigma(t,X_t^{s,x})]^*(\nabla_y u^\infty)(t,X_t^{s,x})) \\ &\frac{[\sigma(s,x)]^*}{t-s}\int_s^t \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^T dW_r\Bigg]dt,\end{aligned} \tag{30}$$

Concatenating the solution as $\mathbf{u}^\infty = (u, \sigma^*\nabla_y u)$, we can define the iteration operator Φ:Lip$([0,T] \times \mathbb{R}^d, \mathbb{R}^{1+d}) \to$ Lip$([0,T] \times \mathbb{R}^d, \mathbb{R}^{1+d})$ as the following

$$\begin{aligned}(\Phi(\mathbf{v}))\,(s,x) =& \mathbb{E}\left[g(X_T^{s,x})\left(1, \frac{[\sigma(s,x)]^*}{T-s}\int_s^T \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right] \\ &+ \int_s^T \mathbb{E}\left[F(\mathbf{v}(t,X_t^{s,x}))\left(1, \frac{[\sigma(s,x)]^*}{t-s}\int_s^t \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right] dt,\end{aligned} \tag{31}$$

and 29, 30 yield

$$\mathbf{u}^\infty = \Phi(\mathbf{u}^\infty). \tag{32}$$

The Multilevel Picard iteration considers simulating the Picard iteartion $\mathbf{u}_k(s,x) = (\Phi(\mathbf{u}_{k-1}))(s,x), k \in \mathbb{N}_+$, which is guaranteed to converge to $\mathbf{u}^\infty$ as $k \to \infty$ for any $s \in [0,T), x \in \mathbb{R}^d$ (Yong and Zhou, 1999, Theorem 7.3.4). Formally, the MLP method uses MLMC(Giles, 2008, 2015) to simulate the following telescope expansion problem derived from the Picard iteration.

$$
\begin{aligned}
\mathbf{u}_k(s,x) &= \mathbf{u}_1(s,x) + \sum_{l=1}^{k-1}\big[\mathbf{u}_{l+1}(s,x) - \mathbf{u}_l(s,x)\big] = \Phi\big(\mathbf{u}_1\big)(s,x) + \sum_{l=1}^{k-1}\Big[\Phi\big(\mathbf{u}_l\big)(s,x) - \Phi\big(\mathbf{u}_{l-1}\big)(s,x)\Big]. \\
&= (g(x),0) + \mathbb{E}\left[(g(X_T^{s,x}) - g(x))\left(1, \frac{[\sigma(s,x)]^*}{T-s}\int_s^T \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right] \\
&+ \sum_{l=0}^{k-1}\int_s^T \mathbb{E}\left[(F(\mathbf{u}_l(t,X_t^{s,x})) - \mathbb{1}_{\mathbb{N}}(l)F(\mathbf{u}_{l-1}(t,X_t^{s,x})))\left(1, \frac{[\sigma(s,x)]^*}{t-s}\int_s^t \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right)\right]dt.
\end{aligned}
\tag{33}
$$

One can either estimate these integrations with the quadrature method(quadrature MLP (E et al., 2021)) or the Monte-Carlo method(full-history MLP (Hutzenthaler et al., 2020b)), detialed intruction is shown in demonstrated in B.2. A comprehensive summary of MLP variants can be found at (Research Group on Stochastic Analysis, University of Duisburg-Essen, 2025).

B.2.1. Implementing Multilevel Picard Iterations

Suppose we are given effective simulators (e.g., Euler–Maruyama or Milstein) parameterized by φ (e.g. discretization level), which produce the numerical approximations

$$
\mathcal{X}_{k,\varphi}^{(l,i)}(s,x,t) \approx X_t^{s,x}, \quad \mathcal{I}_{k,\varphi}^{(l,i)}(s,x,t) \approx \left(1, \frac{[\sigma(s,x)]^*}{t-s}\int_s^t \left[\sigma(r,X_r^{s,x})^{-1}D_r^{s,x}\right]^* dW_r\right), \tag{34}
$$

where k denotes the total level, l the current level, and i (which may be negative) indexes the sample path. To implement the Multilevel Picard Iterations, we need a numerical approximation to the integral $\int_s^T \mathbb{E}F(\mathbf{u}_l(t,X_t^{s,x}))dt$. Following (E et al., 2021; Hutzenthaler et al., 2021), we examine the following two methodologies, using quadrature rule and Monte Carlo algorithm to approximate the integral $\int_s^T \mathbb{E}F(\mathbf{u}_l(t,X_t^{s,x}))dt$:

Quadrature MLP In this approach (E et al., 2021), quadrature rules are employed to approximate the time integrals that appear in the MLP formulation. This quadrature-based technique is motivated by the need to efficiently and accurately resolve time integration errors while maintaining the stability of the multilevel scheme. By leveraging well-established quadrature polynomials, we obtain a deterministic and high-order accurate approximation that is well-suited to the recursive structure of the SCaSML algorithm.

Definition 1 (Quadrature Polynomials). *For each $n \in \mathbb{N}$, let $(c_i^n)_{i=1}^n \subseteq [-1,1]$ denote the n distinct roots of the Legendre polynomial $x \mapsto \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right]$, and define the function $q^{n,[a,b]} : [a,b] \to \mathbb{R}$ by*

$$
q^{n,[a,b]}(t) = \begin{cases} \displaystyle\int_a^b \prod_{\substack{i=1,\ldots,n \\ c_i^n \neq \frac{2t-(a+b)}{b-a}}} \frac{2x-(b-a)c_i^n-(a+b)}{2t-(b-a)c_i^n-(a+b)}\,dx, & \text{if } a<b \text{ and } \frac{2t-(a+b)}{b-a} \in \{c_1^n,\ldots,c_n^n\}, \\ 0, & \text{otherwise.} \end{cases} \tag{35}
$$

The quadrature polynomials serve as a fundamental building block to discretize the time variable in the Picard iteration. With these polynomials, one can approximate the time integrals with high-order accuracy while controlling the error propagation in the recursive iterations.

Definition 2 (Quadrature Multilevel Picard Iteration). *Let* $\left\{\mathbf{U}_{n,M,Q}^{(l,j)}\right\}_{l,j\in\mathbb{Z}} \subseteq \mathcal{M}\Big(\mathcal{B}([0,T]\times\mathbb{R}^d)\otimes \mathcal{F}, \mathcal{B}(\mathbb{R}\times\mathbb{R}^d)\Big)$ *be a family of measurable functions satisfying, for all* $l, j \in \mathbb{N}$ *and* $(s,x)\in[0,T)\times\mathbb{R}^d$*, we start with* $\mathbf{U}_{n,M,Q}^{(0,\pm j)}(s,x) = \mathbf{0}$*. For* $n > 0$*, we define the quadrature SCaSML iteration as*

$$\begin{aligned}\mathbf{U}_{n,M,Q}(s,x) = \Big(g(x),0\Big) + \frac{1}{M^n}\sum_{i=1}^{M^n}\Big(g\big(\mathcal{X}_{k,\varphi}^{(0,-i)}(s,x,T)\big) - g(x)\Big)\mathcal{I}_{k,\varphi}^{(0,-i)}(s,x,T)\\ +\sum_{l=0}^{n-1}\sum_{t\in(s,T)}\frac{q^{Q,[s,T]}(t)}{M^{n-l}}\sum_{i=1}^{M^{n-l}}\Big(F\big(\mathbf{U}_{n,M,Q}^{(l,i)}(t,\mathcal{X}_{k-l,\varphi}^{(l,i)}(s,x,t))\big) - \mathbf{1}_{\mathbb{N}}(l)\,F\big(\mathbf{U}_{n,M,Q}^{(l-1,-i)}(t,\mathcal{X}_{k-l,\varphi}^{(l,i)}(s,x,t))\big)\Big)\\ \cdot\,\mathcal{I}_{k-l,\varphi}^{(l,i)}(s,x,t).\end{aligned} \tag{36}$$

The use of quadrature in this context is motivated by its ability to yield a systematic error control over the temporal discretization, thereby enhancing the stability and accuracy of the multilevel Picard iteration in the simulation-calibrated framework.

Full-history MLP The full-history MLP scheme (Hutzenthaler et al., 2021) adopts a Monte Carlo approach to approximate the time integral $\int_s^T \mathbb{E}F(\mathbf{u}_l(t,X_t^{s,x}))dt$ instead of deterministic quadrature rules with fixed time grids. This modification considerably simplifies error analysis(Hutzenthaler et al., 2020a) and avoids all temporal discretization error.

In the full-history MLP, we employ a time-sampler that guarantees an unbiased Monte Carlo approximation of time integrals. Let $\mathfrak{r}:\Omega\to(0,1)$ be a collection of independent and identically distributed random variables with density ρ satisfying $\mathbb{P}\Big(\mathfrak{r}^{(l,i)}\leqslant b\Big) = \int_0^b\rho(s)\,ds$. Consider numerically approximating the integral $I(f;s,t) = \int_s^t f(r)\,dr$ with $t\in(s,T)$, we construct an importance sampling estimator with sample size N:$\hat{I}(f;s,t) = \frac{1}{N}\sum_{i=1}^N \frac{f(R^{(i)})\,\mathbb{1}_{\{R^{(i)}\leq t\}}}{\varrho(R^{(i)},s)}$, where ϱ is the the rescaled density ρ on (s,T) defined as $\varrho(r,s) = \frac{\rho\left(\frac{r-s}{T-s}\right)}{T-s}$ and R is the random sample from the density $\varrho(\cdot,s)$ on (s,T) via $R = s+(T-s)\,\mathfrak{r}$.

Definition 3 (Full-history Multilevel Picard Iteration (Hutzenthaler et al., 2020a)). *Let* $\left\{\mathbf{U}_{n,M}^{(l,j)}\right\}_{l,j\in\mathbb{Z}} \subseteq \mathcal{M}\Big(\mathcal{B}([0,T]\times\mathbb{R}^d)\otimes\mathcal{F},\mathcal{B}(\mathbb{R}\times\mathbb{R}^d)\Big)$ *be a family of measurable functions satisfying, for all* $l,j\in\mathbb{N}$ *and* $(s,x)\in[0,T)\times\mathbb{R}^d$*, we start with* $\mathbf{U}_{n,M}^{(0,\pm j)}(s,x) = \mathbf{0}$*. Then, for* $n > 0$*, define the full-history SCaSML iteration as*

$$\begin{aligned}\mathbf{U}_{n,M}(s,x) = \Big(g(x),0\Big) + \frac{1}{M^n}\sum_{i=1}^{M^n}\Big(g\big(\mathcal{X}_{k,\varphi}^{(0,-i)}(s,x,T)\big) - g(x)\Big)\mathcal{I}_{k,\varphi}^{(0,-i)}(s,x,T)\\ +\sum_{l=0}^{n-1}\frac{1}{M^{n-l}}\sum_{i=1}^{M^{n-l}}\frac{1}{\varrho(s,\mathcal{R}_s^{(l,i)})}\Big(F\big(\mathbf{U}_{n,M}^{(l,i)}(\mathcal{R}_s^{(l,i)},\mathcal{X}_{k-l,\varphi}^{(l,i)}(s,x,\mathcal{R}_s^{(l,i)}))\big)\\ -\mathbf{1}_{\mathbb{N}}(l)F\big(\mathbf{U}_{n,M}^{(l-1,-i)}(\mathcal{R}_s^{(l,i)},\mathcal{X}_{k-l,\varphi}^{(l,i)}(s,x,\mathcal{R}_s^{(l,i)}))\big)\Big)\cdot\mathcal{I}_{k-l,\varphi}^{(l,i)}\Big(s,x,\mathcal{R}_s^{(l,i)}\Big),\end{aligned} \tag{37}$$

here $\mathcal{R}_s^{(l,i)}$ *is i-th sampled time point after t at level l which is defined as as* $\mathcal{R}_s^{(l,i)} = s+(T-s)\,\mathfrak{r}^{(l,i)}$.

C. Algorithm

In this section, we describe the complete procedure of Simulation-Calibrated Scientific Machine Learning (SCaSML) for solving high-dimensional partial differential equations (1). The SCaSML framework at any space-time point (t, x) can be summarized as follows:

- **Step 1:Train a Base Surrogate.** First, a surrogate model $\hat{u}$ is trained to approximately solve the target PDE (1), serving as a preliminary estimate of the true solution.
- **Step 2:Physics-Informed Inference-Time Scaling via the Law of Defect.** Recognizing that the defect $\breve{u} := u - \hat{u}$ satisfies a semi-linear parabolic equation, termed the Law of defect,

$$\begin{cases} \frac{\partial}{\partial r}\breve{u} + \langle \mu, \nabla_y \breve{u} \rangle + \frac{1}{2}\operatorname{Tr}\big(\sigma^* \ \operatorname{Hess}_y \breve{u}\, \sigma\big) + \breve{F}\big(\breve{u}, \sigma^* \nabla_y \breve{u}\big) = 0, & \text{on } [0, T) \times \mathbb{R}^d, \\ \breve{u}(T, y) = \breve{g}(y), & \text{on } \mathbb{R}^d, \end{cases}$$

  one obtains an estimate of $\breve{u}(t, x)$ by employing Multilevel Picard iteration, either through quadrature-based MLP (Definition 2) or full-history MLP (Definition 3).
- **Step 3:Final Estimation.** The final estimate of the solution is then given by $u(t, x) \approx \hat{u}(t, x) + \breve{u}(t, x)$.

The entire algorithm is detailed in Algorithm 1.

Remark 1. *We emphasize that the sample-wise iteration in Algorithm 1 can be substituted by vectorized operations, thereby enabling the algorithm to be applied concurrently to multiple points. These performance enhancements were implemented using `JAX` and `DeepXDE`, resulting in a time reduction by a factor of* $5\times$ *to* $10\times$.

Remark 2. *Additionally, methods such as thresholding (Sebastian Becker et al., 2020) and Hutchinson's estimator (Hutchinson, 1989; Shi et al., 2025) could also be employed within the principal algorithm. Thresholding (Algorithm 2) mitigates numerical instability by methodically "clipping" the defect estimator* $\breve{\mathbf{U}}$*, a critical action when the surrogate model yields outlier values or when unbounded growth may manifest during iterative correction phases. Hutchinson's estimator (Algorithm 3) alleviates the computational and memory demands of* ε_{PDE} *in* $\breve{F}$ *by forming an unbiased estimator that necessitates only a subset of second-order derivatives approximating the Laplacian. This partial evaluation not only expedites the simulation process but also minimizes peak memory consumption, thus averting out-of-memory issues.*

D. Assumptions on Surrogate Models

In the following sections, we derive the main results of our work. Specifically, we show that the global L^2 error/computational complexity for SCaSML to achieve a given precision can be expressed as the product of (i) the global L^2 error/computational complexity of the MLP for achieving the same error and (ii) the global L^2 error of the surrogate model $\hat{u}$.

We begin by introducing the notations and the regularity assumptions, which apply to both MLP models. The proofs for the quadrature MLP and full-history MLP are developed independently. Note that, since we use a generic error function $e(\hat{u})$ with minimal constraints, separate proof validations for each surrogate are not required.

D.1. Notations

Our analysis follows the structure in (E et al., 2021; Hutzenthaler et al., 2020a). In our setting, we adopt the following definitions.

Algorithm 1 Simulation-Calibrated Scientific Machine Learning for Solving High-Dimensional Partial Differential Equation

```
Require: Level $n$, sample base $M$, target point $(s,x)$, a surrogate model $\hat{u}$, threshold $\varepsilon$, (quadrature order $Q$ for using Qudrature MLP)
Train a base surrogate model $\hat{u}$ to approximate the PDE solution.
Take MLP_Law_of_Defect$(s,x,n,M,Q) \cdot (1,\vec{\mathbf{0}}) + \hat{u}(s,x)$ as estimation of $u(s,x)$
function MLP_LAW_OF_DEFECT$(s,x,n,M,Q,\hat{u})$
    $\hat{\mathbf{u}}(s,x) \leftarrow \Big(\hat{u}(s,x),\ \sigma^*(s,x)\nabla_y\hat{u}(s,x)\Big)$
    if $n=0$ then                ▷ Start Inference-Time Scaling via Simulating the law of defect
        $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow \vec{\mathbf{0}}$
        return $\breve{\mathbf{U}}_{n,M,Q}(s,x)$
    end if
    $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow (\breve{g}(x),0)$
    for $i=1$ to $M^n$ do
        Sample the Feyman-Kac Path $\mathcal{X}^{(0,-i)}_{k,\varphi}(s,x,T)$ and Derivative process $\mathcal{I}^{(0,-i)}_{k,\varphi}(s,x,T)$ according to (34)
        $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow \breve{\mathbf{U}}_{n,M,Q}(s,x) + \frac{1}{M^n}\Big(\breve{g}\big(\mathcal{X}^{(0,-i)}_{k,\varphi}(s,x,T)\big) - \breve{g}(x)\Big)\cdot\mathcal{I}^{(0,-i)}_{k,\varphi}(s,x,T)$
    end for
    for $l=0$ to $n-1$ do
        for $i=1$ to $M^{n-l}$ do
            if using Quadrature MLP to calibrate then
                Compute $Q$ quadrature points and their corresponding weights $q^{Q,[s,T]}(t)$ with 1
                for all quadrature points $t\in[s,T]$ do
                    Sample the Feyman-Kac Path $\mathcal{X}^{(l,i)}_{k,\varphi}(s,x,t)$ and Derivative process $\mathcal{I}^{(l,i)}_{k,\varphi}(s,x,t)$ according to (34)
                    $\mathbf{z} \leftarrow \breve{\mathbf{U}}^{(l,i)}_{n,M,Q}\big(t,\mathcal{X}^{(l,i)}_{k-l,\varphi}(s,x,t)\big)$
                    if $l>0$ then
                        $\mathbf{z}_{\text{prev}} \leftarrow \breve{\mathbf{U}}^{(l-1,-i)}_{n,M,Q}\big(t,\mathcal{X}^{(l,i)}_{k-l,\varphi}(s,x,t)\big)$, $\Delta\breve{F} \leftarrow \breve{F}(\mathbf{z}) - \breve{F}(\mathbf{z}_{\text{prev}})$
                    else
                        $\Delta\breve{F} \leftarrow \breve{F}(\mathbf{z})$
                    end if
                    $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow \breve{\mathbf{U}}_{n,M,Q}(s,x) + \frac{q^{Q,[s,T]}(t)}{M^{n-l}}\ \Delta\breve{F}\cdot\mathcal{I}^{(l,i)}_{k-l,\varphi}(s,x,t)$
                end for
            end if
            if using Full History MLP to calibrate then
                Sample time step $\mathcal{R}^{(l,i)}_s \sim \varrho(s,T)$
                Sample the Feyman-Kac Path $\mathcal{X}^{(l,i)}_{k,\varphi}(s,x,\mathcal{R}^{(l,i)}_s)$ and Derivative process $\mathcal{I}^{(l,i)}_{k,\varphi}(s,x,\mathcal{R}^{(l,i)}_s)$ according to (34)
                $\mathbf{z} \leftarrow \breve{\mathbf{U}}^{(l,i)}_{n,M,Q}\big(\mathcal{R}^{(l,i)}_s,\mathcal{X}^{(l,i)}_{k-l,\varphi}(s,x,\mathcal{R}^{(l,i)}_s)\big)$
                if $l>0$ then
                    $\mathbf{z}_{\text{prev}} \leftarrow \breve{\mathbf{U}}^{(l-1,-i)}_{n,M,Q}\big(\mathcal{R}^{(l,i)}_s,\mathcal{X}^{(l,i)}_{k-l,\varphi}(s,x,\mathcal{R}^{(l,i)}_s)\big)$, $\Delta\breve{F} \leftarrow \breve{F}(\mathbf{z}) - \breve{F}(\mathbf{z}_{\text{prev}})$
                else
                    $\Delta\breve{F} \leftarrow \breve{F}(\mathbf{z})$
                end if
                $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow \breve{\mathbf{U}}_{n,M,Q}(s,x) + \frac{1}{M^{n-l}}\cdot\frac{1}{\varrho(s,\mathcal{R}^{(l,i)}_s)}\cdot\Delta\breve{F}\cdot\mathcal{I}^{(l,i)}_{k-l,\varphi}(s,x,\mathcal{R}^{(l,i)}_s)$
            end if
        end for
    end for
    $\breve{\mathbf{U}}_{n,M,Q}(s,x) \leftarrow$ Thresholding$(\varepsilon,\breve{\mathbf{U}}_{n,M,Q}(s,x))$        ▷ Threshold outliers using Algorithm 2
    return $\breve{\mathbf{U}}_{n,M,Q}(s,x)$
end function
```

Algorithm 2 Thresholding the outliers (Sebastian Becker et al., 2020)

```
Require: Threshold $\varepsilon$, defect estimator $\check{\mathbf{U}}$
function THRESHOLDING($\varepsilon$, $\check{\mathbf{U}}$)
    for $\nu = 1$ to $d + 1$ do
        if $\check{\mathbf{U}}_\nu > \varepsilon$ then
            $\check{\mathbf{U}}_\nu \longleftarrow \varepsilon$
        end if
        if $\check{\mathbf{U}}_\nu < -\varepsilon$ then
            $\check{\mathbf{U}}_\nu \longleftarrow -\varepsilon$
        end if
    end for
    return Clipped $\check{\mathbf{U}}$
end function
```

Algorithm 3 Hutchison's estimator for estimating Laplacian (Shi et al., 2025)

```
Require: Sample size $K$, target function $f$
function HTE($K$,$f$)
    Draw $K$ different indices from $1, \ldots, d$ with equal probability $1/d$, denoted as $j_1, \ldots, j_K$
    Compute $D^2_{j_k} f, 1 \leqslant \nu \leqslant d$
    Compute estimator HTE $\longleftarrow \frac{d}{K} \sum_{i=1}^{K} D^2_{j_i} f$
    return Laplacian estimator HTE
end function
```

Definition 1 (Discrete Inner Product and Norm). *For each $n \in \mathbb{N}$ and vectors $v = (v_1, \ldots, v_n)$, $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$, the discrete inner product is defined as $\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$. For $p \in \mathbb{N}$, the discrete p-norm is defined by $\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}}$, and the discrete ∞-norm by $\|v\|_\infty = \max_{1 \le i \le n} |v_i|$.*

Definition 2 (Probability Space and Continuous Norm). *Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. For a measurable function $X : \Omega \to \mathbb{R}$ and $p \in \mathbb{N}$, the L^p-norm is defined as $\|X\|_{L^p} = \left(\mathbb{E}[|X|^p]\right)^{\frac{1}{p}}$, and the L^∞-norm as $\|X\|_{L^\infty} = ess\ sup_{\omega \in \Omega} |X(\omega)|$.*

Definition 3 (Sobolev Space and Norm). *For a probability space Ω, $k \in \mathbb{N}$, and $1 \le p \le \infty$, the Sobolev space $W^{k,p}(\Omega)$ consists of functions $u \in L^p(\Omega)$ whose weak derivatives $D^\alpha u$ exist and belong to $L^p(\Omega)$ for all multi-indices α with $|\alpha| \le k$. Its norm is given by*

$$
\|u\|_{W^{k,p}(\Omega)} = \begin{cases} \sum_{|\alpha| \le k} \|D^\alpha u\|_{L^p(\Omega)}, & 1 \le p < \infty, \\ \sum_{|\alpha| \le k} \|D^\alpha u\|_{L^\infty(\Omega)}, & p = \infty. \end{cases} \tag{38}
$$

Definition 4 (Laws of Extended Real Numbers). *We adopt the following conventions:$\frac{0}{0} = 0$, $0 \cdot \infty = 0$, $0^0 = 1$, and $\sqrt{\infty} = \infty$. For every $a > 0$ and $b \in \mathbb{R}$, we define $\frac{a}{0} = \infty$, $\frac{-a}{0} = -\infty$, $0^{-a} = \infty$, $\frac{1}{0^a} = \infty$, $\frac{b}{\infty} = 0$, and $0^a = 0$.*

D.2. Regularity Assumptions for Surrogate Model

Let $T \in (0, \infty)$, $d \in \mathbb{N}$, $F : \mathcal{M}(\mathcal{B}([0,T] \times \mathbb{R}^d), \mathcal{B}(\mathbb{R}^{d+1})) \to \mathcal{M}(\mathcal{B}([0,T] \times \mathbb{R}^d), \mathcal{B}(\mathbb{R}))$, $g \in C^2(\mathbb{R}^d, \mathbb{R})$, and $L \in \mathbb{R}^{d+1}, K \in \mathbb{R}^d$. Let $(\Omega, \mathcal{F}, \mathbb{P}, (\mathbb{F}_t)_{t \in [0,T]})$ be a stochastic basis. Assume that $W^{(l,j)} : [0,T] \times \Omega \to \mathbb{R}^d$ for $l, j \in \mathbb{Z}$ are independent standard Brownian motions, where the index l represents the level in the MLP algorithm and j differentiates between positive and negative samples for error correction.

In the proof, we will maintain notations for PDE as we did in the main text, but setting $\mu = \mathbf{0}, \sigma \in \mathbb{R} \cdot \mathbf{I}$. In addition, for simplicity, we will see σ as a real number in the proof.

We use subscripts (typically ν or α) to denote individual components of vectors.

Now we list regularity assumptions for our PDE model.

Assumption 5 (Lipschitz Continuity of Nonlinearity and Terminal). *For all $\mathbf{u}_1, \mathbf{u}_2$ in $\mathcal{M}(\mathcal{B}([0,T] \times \mathbb{R}^d), \mathcal{B}(\mathbb{R}^{d+1}))$ and for all $(r,y) \in [0,T] \times \mathbb{R}^d$, there exist constants $L_\nu > 0$ ($\nu = 1, \ldots, d+1$) such that*

$$\left|(F(\mathbf{u}_1) - F(\mathbf{u}_2))(r,y)\right| \leq \sum_{\nu=1}^{d+1} L_\nu \left|\left(\mathbf{u}_1(r,y) - \mathbf{u}_2(r,y)\right)_\nu\right|, \tag{39}$$

and the terminal function $g : \mathbb{R}^d \to \mathbb{R}$ satisfies, for all $x, y \in \mathbb{R}^d$,

$$|g(x) - g(y)| \leq \sum_{\alpha=1}^{d} K_\alpha |(x-y)_\alpha|, \tag{40}$$

for some constants $K_\alpha > 0$.

In practice, we assume the existence of a surrogate model parameterized by θ (which may be biased, e.g. PINN, Gaussian Process, etc.) that produces an estimated solution $\hat{u} \in W^{1,\infty}([0,T] \times \mathbb{R}^d)$. Associated with this surrogate is an error controller $e(\hat{u})$. The goal of SCaSML is to compute the bias $\breve{u} = u^\infty - \hat{u}$. The assumptions regarding the surrogate model's accuracy (assumption 1 or 1) are specified for each correction method in the corresponding proof sections.

Assumption 6 (Lipschitz Continuity of the Surrogate Terminal). *Assume that there exists $\hat{K} \in \mathbb{R}^d$ such that $|\hat{g}(x) - \hat{g}(y)| \leq \sum_{\alpha=1}^{d} \hat{K}_\alpha \, |(x-y)_\alpha|, \quad \forall x, y \in \mathbb{R}^d$.*

We can easily show that the Lipschitz conditions 5 and 6 transfer to bias:

Definition 7 (Lipschitz Continuity of the Bias Nonlinearity and Terminal Condition). *Assume that assumption 6 holds. We define $\breve{L} \in \mathbb{R}^{d+1}$ and $\breve{K} \in \mathbb{R}^d$ as vectors of minimal $\|\cdot\|_1$ norm such that for all $(r,y) \in [0,T] \times \mathbb{R}^d$,*

$$\left|\breve{F}(\breve{\mathbf{u}}_1)(r,y) - \breve{F}(\breve{\mathbf{u}}_2)(r,y)\right| \leq \sum_{\nu=1}^{d+1} \breve{L}_\nu \left|(\breve{\mathbf{u}}_1(r,y) - \breve{\mathbf{u}}_2(r,y))_\nu\right|, \tag{41}$$

and for all $x, y \in \mathbb{R}^d$,

$$|\breve{g}(x) - \breve{g}(y)| \leq \sum_{\alpha=1}^{d} \breve{K}_\alpha \, |(x-y)_\alpha|, \tag{42}$$

as the Lipschitz constants for $\breve{F}$ and $\breve{g}$.

Remark 8. *We observe that $\left|\breve{F}(\breve{\mathbf{u}}_1)(r,y) - \breve{F}(\breve{\mathbf{u}}_2)(r,y)\right| = |F(\hat{\mathbf{u}} + \breve{\mathbf{u}}_1)(r,y) - F(\hat{\mathbf{u}})(r,y) - F(\hat{\mathbf{u}} + \breve{\mathbf{u}}_2)(r,y) + F(\hat{\mathbf{u}})(r,y)| = |F(\hat{\mathbf{u}} + \breve{\mathbf{u}}_1)(r,y) - F(\hat{\mathbf{u}} + \breve{\mathbf{u}}_2)(r,y)| \leqslant \sum_{\nu=1}^{d+1} L_\nu |(\breve{\mathbf{u}}_1(r,y) - \breve{\mathbf{u}}_2(r,y))_\nu|$. This directly implies that $\|\breve{L}\|_1 \leqslant \|L\|_1$, providing a moderate upper bound for the Lipschitz constant associated with the bias term. This bound is instrumental in the analyses of theorems 4 and 2, reinforcing the reasonableness of the corresponding error estimates. Similarly, we also have an initial estimation $\breve{K}_\alpha \leqslant K_\alpha + \hat{K}_\alpha$ for all $\alpha = 1, \ldots, d$, which derives $\|\breve{K}\|_1 \leqslant \|K\|_1$.*

E. Proof for Quadrature Multilevel Picard Iteration

In this section, we present the proof for the Quadrature Multilevel Picard (MLP) iteration method. For simplicity, we consider the case where $\mu = 0$ and $\sigma = s\mathbf{I}_d (s \in \mathbb{R})$ in the proof. We first establish the mathematical framework and underlying assumptions, then analyze the convergence properties and computational complexity of our proposed simulation-calibrated variant. The result shows that the error of SCaSML is bounded by the product of MLP error and surrogate error. Likewise, the complexity is bounded by the product of MLP error and surrogate error. Both indicate that surrogate models can substantially reduce computational complexity while maintaining accuracy guarantees.

Since the `Law of defect` is also a semi-linear heat equation, we can use the quadrature/full-history multilevel Picard iteration to obtain an estimation $\breve{\mathbf{U}}(s,x)$ of $u(s,x) - \hat{u}(s,x)$. In this section, we study the theoretical properties of SCaSML that using Quadrature Multilevel Picard Iteration to solve the `Law of defect` and we investigate the full-history multilevel Picard iteration in the next section.

E.1. Assumptions of the Surrogate Model

Inference-time scaling of a language model demands a strong enough base language model. Similarly, scaling a neural PDE solver at inference time requires an accurately approximated surrogate model. In this section, we elucidate the assumptions embedded in the error measure $e(\hat{u})$—a metric that quantifies the surrogate model's accuracy and underpins the enhanced convergence rates achieved by the SCaSML methodology.

Assumption 1 (Accuracy of the Surrogate Model for Quadrature MLP)**.** *Let $\breve{u}^\infty \in C^\infty([0,T] \times \mathbb{R}^d, \mathbb{R})$ and $\sup_{t\in[0,T]} \|\breve{u}^\infty(t,\cdot)\|_{W^{1,\infty}} < \infty$. There exist constants $C_{Q,1}, C_{Q,2}, C_{Q,3} > 0$ such that:*

1. *L^∞ PDE Residual Bound:*$\sup_{r\in[0,T], y\in\mathbb{R}^d} |\epsilon_{PDE}(r,y)| \leqslant C_{Q,1}\, e(\hat{u})$
2. *$W^{1,\infty}$ Error Bound:*$\sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{Q,2}\, e(\hat{u})$
3. *Smoothness of the solution:*$\sup_{k\in\mathbb{N}} \dfrac{\|(1,\nabla_y)\Big((\frac{\partial}{\partial r} + \frac{\sigma^2}{2}\Delta_y)^k \breve{u}\Big)(r,y)\|_{L^\infty}}{(k!)^{3/4}} \leqslant C_{Q,3}\, e(\hat{u})$

Assumption 2 (Quadrature Integrability Assumption (Hutzenthaler and Kruse, 2020, Lemma 4.1))**.** *There exists $p \in \mathbb{N}$ such that for the zero function $\mathbf{0}$ (initializer) and for all $t \in [0,T]$,*

$$\sup_{x\in\mathbb{R}^d} \frac{|\breve{g}(x)|}{1 + \|x\|_1^p} + \sup_{x\in\mathbb{R}^d} \frac{|\breve{F}(\mathbf{0})(t,x)|}{1 + \|x\|_1^p} < \infty. \tag{43}$$

This accuracy of the surrogate model establishes a critical relationship between several error terms and our error measure $e(\hat{u})$. Specifically, it ensures that the PDE residual, the bias of the surrogate model θ, and the amplification resulting from repeated applications of the PDE operator are all controlled by $e(\hat{u})$.

This assumption guarantees that both the terminal condition and the nonlinearity exhibit suitable growth properties, which is essential for ensuring that our numerical scheme remains stable.

E.2. Main Results

We now present our main theoretical results, which characterize both the accuracy and computational complexity of our proposed method. These results demonstrate the substantial efficiency gains achieved by incorporating surrogate models into the multilevel Picard framework.

E.2.1. Bound on Global L^2 Error

Our proof leverages the observation that the global L^2 error of the MLP is primarily determined by the Lipschitz continuity of the PDE's terminal&solution and the magnitude of its nonlinearity at the origin. We demonstrate that the factor associated with the `Law of defect` are bounded by the surrogate error. We first show how the complexity characterization of MLP can be bounded by the error measurement in the following lemma.

Lemma 3 (Complexity Estimation via Surrogate Model's Error Measure $e(\hat{u})$)**.** *Suppose assumptions 5, 6, 2 and 1 hold. There exists a constant $C_Q > 0$ such that*

$$\sup_{(r,y)\in[0,T]\times\mathbb{R}^d} \left\{ \left|(\breve{F}(\mathbf{0}))(r,y)\right| + \sigma\sqrt{T+3}\,\|\breve{K}\|_1 \quad + \sup_{k\in\mathbb{N}} \frac{\|(1,\nabla_y)\Big((\frac{\partial}{\partial r} + \frac{\sigma^2}{2}\Delta_y)^k \breve{u}^\infty\Big)(r,y)\|_\infty}{(k!)^{3/4}} \right\} \leqslant C_Q\, e(\hat{u}). \tag{44}$$

Proof. To establish this bound, we analyze each term separately using the accuracy of the surrogate model (Assumption 1). For the first term, recalling that $\breve{F}(\mathbf{v}) := F((\hat{u}, \sigma\nabla_y\hat{u}) + \mathbf{v}) - F((\hat{u}, \sigma\nabla_y\hat{u})) + \varepsilon_{PDE} \Rightarrow \breve{F}(\mathbf{0}) = \varepsilon_{PDE}$, we can directly apply L^∞ residual bound in assumption 1:

$$\sup_{(r,y)\in[0,T]\times\mathbb{R}^d} |(\breve{F}(\mathbf{0}))(r,y)| = \sup_{r\in[0,T],y\in\mathbb{R}^d} |\varepsilon_{\mathrm{PDE}}(r,y)| \leqslant C_{Q,1}e(\hat{u}). \tag{45}$$

Suppose e_α is the unit vector with 1 at its α-th component. For the second term, we note that $\breve{K}$ is the vector of Lipschitz constants of $\breve{g}$, so it must holds that $\breve{K}_\alpha \leqslant \|D^{e_\alpha}\breve{g}\|_{L^\infty}$ (Just check the definition in 7) $\Rightarrow$ $\|\breve{K}\|_1 = \sum_{\alpha=1}^d \breve{K}_\alpha \leqslant \sum_{\alpha=1}^d \|D^{e_\alpha}\breve{g}\|_{L^\infty} = \sum_{\alpha=2}^{d+1} \|D^{e_\alpha}u(T,\cdot)\|_{L^\infty}$, and the RHS can be controlled by $W^{1,\infty}$ error bound in assumption 1. Combing this fact and definition of Sobolev norm 3, we get:

$$\|\breve{K}\|_1 \leqslant \sum_{\alpha=1}^{d} \|D^{e_\alpha}\breve{g}\|_{L^\infty} \leqslant \sum_{\alpha=2}^{d+1} \|D^{e_\alpha}\breve{u}^\infty(T,\cdot)\|_{L^\infty} + \|\breve{u}^\infty(T,\cdot)\|_{L^\infty} = \|\breve{u}(T,\cdot)\|_{W^{1,\infty}} \tag{46}$$

$$\leqslant \sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{Q,2}e(\hat{u}). \tag{47}$$

For the third term, we observe that L^∞ operator bound in assumption 1 is identical to the conclusion, which gives:

$$\sup_{k\in\mathbb{N}} \frac{\|(1,\nabla_y)\Big((\frac{\partial}{\partial r}+\frac{\sigma^2}{2}\Delta_y)^k\breve{u}^\infty\Big)(r,y)\|_{L^\infty}}{(k!)^{\frac{3}{4}}} \leqslant C_{Q,3}e(\hat{u}). \tag{48}$$

Plugging 45, 46 and 48 in the LHS of 3, we have

$$\sup_{(r,y)\in[0,T]\times\mathbb{R}^d} \left\{ \left|(\breve{F}(\mathbf{0}))(r,y)\right| + \sigma\sqrt{T+3}\,\|\breve{K}\|_1 \quad + \sup_{k\in\mathbb{N}} \frac{\|(1,\nabla_y)\Big((\frac{\partial}{\partial r} + \frac{\sigma^2}{2}\Delta_y)^k \breve{u}^\infty\Big)(r,y)\|_\infty}{(k!)^{3/4}} \right\} \tag{49}$$

$$\leqslant (C_{Q,1} + C_{Q,2}\sigma\sqrt{T+3} + C_{Q,3})e(\hat{u}). \tag{50}$$

Defining $C_Q = C_{Q,1} + C_{Q,2}\sigma\sqrt{T+3} + C_{Q,3}$ in 49, we complete the proof. □

This lemma is instrumental in establishing our main error bound, as it directly connects the key parameters of our numerical scheme to the surrogate model's quality measure $e(\hat{u})$.

Theorem 4 (Bound of Global L^2 Error)**.** *Under assumptions 5, 6, 2 and 1, it holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\breve{\mathbf{U}}_{N,N,N}(t,x) - \breve{\mathbf{u}}^{\infty}(t,x) \right)_{\nu} \right\|_{L^2} \leqslant C_Q\, e(\hat{u})\; E(N), \tag{51}$$

where

$$E(N) = \frac{7C^N\, 2^{N-1} e^N}{\sqrt{N^{N-3}}} + \frac{\left(14(4C)^{N-1} + 1\right) T^{2N+1}}{\sqrt{N^N}}, \tag{52}$$

and

$$C = 2(\sqrt{T}+1)\sqrt{T\pi}(\|\breve{L}\|_1 + 1) + 1. \tag{53}$$

Proof. Under assumptions 2 and 7, (Hutzenthaler and Kruse, 2020, Corollary 4.7) indicates that for $n = M = Q = N \geqslant 2$, we obtain an error bound of the form:

$$\begin{aligned} \left\| \left(\breve{\mathbf{U}}_{N,N,N}(t,x) - \breve{\mathbf{u}}^{\infty}(t,x) \right)_{\nu} \right\|_{L^2} \leqslant & E(N) \cdot \sup_{(r,y)\in[0,T]\times\mathbb{R}^d} \left\{ \left| (\breve{F}(\mathbf{0}))(r,y) \right| + \sigma\sqrt{T+3}\, \|\breve{K}\|_1 \right. \\ & \left. + \sup_{k\in\mathbb{N}} \frac{\|(1,\nabla_y)\left((\frac{\partial}{\partial r} + \frac{\sigma^2}{2}\Delta_y)^k \breve{u}^{\infty} \right)(r,y)\|_{\infty}}{(k!)^{3/4}} \right\}. \end{aligned} \tag{54}$$

Applying Lemma 3 to substitute the supremum term in 54 with $C_Q\, e(\hat{u})$ completes the proof. □

This theorem provides a comprehensive characterization of the approximation error for our method. The error bound consists of two components:(1) the surrogate model error measure $e(\hat{u})$ scaled by a constant C_Q, and (2) a term $E(N)$ that depends on the number of iterations N and decreases as N increases. This factorization clearly demonstrates how the quality of the surrogate model directly influences the overall approximation error.

An error order estimation w.r.t. N immediately follows:

Corollary 5 (Error Order)**.** *Under assumptions 5, 6, 2 and 1, it holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\breve{\mathbf{U}}_{N,N,N}(t,x) - \breve{\mathbf{u}}^{\infty}(t,x) \right)_{\nu} \right\|_{L^2} \leqslant C_Q\, e(\hat{u})\; \exp\left(N \log N(-\frac{1}{2} + o(1)) \right), \tag{55}$$

Proof. It suffices to estimate the order of $E(N)$. Note that for the first term

$$\begin{aligned} \frac{7C^N\, 2^{N-1} e^N}{\sqrt{N^{N-3}}} &= \exp\left(\log 7 + N\log C + (N-1)\log 2 + N - \frac{N-3}{2}\log N \right) \qquad (56) \\ &= \exp\left(N\log N(-\frac{1}{2} + o(1)) \right). \qquad (57) \end{aligned}$$

Similarly, for the second term

$$\frac{\left(14(4C)^{N-1}+1\right)T^{2N+1}}{\sqrt{N^N}} = \exp\left((2N+1)\log T + \log(14(4C)^{N-1}+1) - \frac{N}{2}\log N\right) \tag{58}$$

$$= \exp\left(N\log N(-\frac{1}{2}+o(1))\right). \tag{59}$$

Thus, combining 56 and 58, we have $E(N) = O\left(\exp\left(N\log N(-\frac{1}{2}+o(1))\right)\right)$. □

E.2.2. Bound on Computational Complexity

To fully assess the efficiency of our method, we now analyze its computational complexity. We introduce two key metrics that capture different aspects of the computational cost.

Definition 6 (Computational Complexity of Quadrature SCaSML). *We define the following complexity measures:*

First, let $\{\mathrm{RN}_{n,M,Q}\}_{n,M,Q\in\mathbb{Z}} \subset \mathbb{N}$ *satisfy* $\mathrm{RN}_{0,M,Q} = 0$ *and, for all* $n, M, Q \in \mathbb{N}$,

$$\mathrm{RN}_{n,M,Q} \leqslant d\,M^n + \sum_{l=0}^{n-1}\Big[Q\,M^{n-l}\Big(d + \mathrm{RN}_{l,M,Q} + \mathbf{1}_{\mathbb{N}}(l)\ \mathrm{RN}_{l-1,M,Q}\Big)\Big]. \tag{60}$$

This number represents the total scalar normal random variable realizations required for computing one sample of $\breve{\mathbf{U}}_{n,M,Q}(s,x)$.

Second, let $\{\mathrm{FE}_{n,M,Q}\}_{n,M,Q\in\mathbb{Z}} \subset \mathbb{N}$ *satisfy* $\mathrm{FE}_{0,M,Q} = 0$ *and, for all* $n, M, Q \in \mathbb{N}$,

$$\mathrm{FE}_{n,M,Q} \leqslant M^n + \sum_{l=0}^{n-1}\Big[Q\,M^{n-l}\Big(1 + \mathrm{FE}_{l,M,Q} + \mathbf{1}_{\mathbb{N}}(l) + \mathbf{1}_{\mathbb{N}}(l)\ \mathrm{FE}_{l-1,M,Q}\Big)\Big]. \tag{61}$$

This quantity reflects the number of evaluations of $\breve{F}$ *and* $\breve{g}$ *necessary to compute of one sample of* $\breve{\mathbf{U}}_{n,M,Q}(s,x)$.

These metrics provide a comprehensive measure of the computational resources required by our method. The first metric, $\mathrm{RN}_{n,M,Q}$, accounts for the cost of generating random variables, while the second, $\mathrm{FE}_{n,M,Q}$, captures the number of function evaluations needed.

Theorem 7 (Complexity of Quadrature SCaSML). *Under assumptions 5, 6, 2 and 1, for any* $\delta > 0$ *and all* $N \in \mathbb{N}$, *we have*

$$\begin{aligned}\mathrm{RN}_{N,N,N} + \mathrm{FE}_{N,N,N} \leq &\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\ \max_{\nu\in\{1,\ldots,d+1\}}\left\|\left(\tilde{\mathbf{U}}_{N,N,N}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(4+\delta)} \\ &\cdot 8(d+1)(C_Q e(\hat{u}))^{4+\delta}\exp\left(N\log N(-\frac{\delta}{2}+o(1))\right) < \infty.\end{aligned} \tag{62}$$

Proof. From established results in (Hutzenthaler et al., 2020a, Lemma 3.6), we know that for all $N \in \mathbb{N}$,

$$\mathrm{RN}_{N,N,N} \leqslant 8dN^{2N}, \quad \mathrm{FE}_{N,N,N} \leqslant 8N^{2N}. \tag{63}$$

We want to use the $O(N^{N/2})$ denominator in Theorem 4 to compensate for the N^{2N} term in the complexity. Suppose the maximum error is ε, and note that $N^{2N} = (N^{N/2})^4 < (N^{N/2})^{4+\delta}, \forall \delta > 0$, we multiply the complexity by $\varepsilon^{4+\delta}$, i.e.

$$\begin{aligned}
&(\mathrm{RN}_{N,N,N} + \mathrm{FE}_{N,N,N}) \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,N,N}(t,x) - \mathbf{u}^\infty(t,x) \right)_\nu \right\|_{L^2} \right]^{(4+\delta)} \\
\leqslant& 8(d+1)N^{2N} \cdot \Bigg(\frac{7\left(2(\sqrt{T}+1)\sqrt{T\pi}(\|\breve{L}\|_1+1)+1\right)^N 2^{N-1}e^N}{\sqrt{N^{N-3}}} \\
&+ \frac{(14(8(\sqrt{T}+1)\sqrt{T\pi}(\|\breve{L}\|_1+1)+4)^{N-1}+1)T^{2N+1}}{\sqrt{N^N}} \Bigg)^{(4+\delta)} (C_Q e(\hat{u}))^{4+\delta} \qquad (64) \\
\leqslant& 8(d+1)N^{2N} \cdot \left((24(T+1))^{3N} \left(\|\breve{L}\|_1 + 1 \right)^N \sqrt{N}^{-N} \right)^{(4+\delta)} (C_Q e(\hat{u}))^{4+\delta} \\
\leqslant& 8(d+1)(C_Q e(\hat{u}))^{4+\delta} \exp\left(N \log N (-\frac{\delta}{2} + o(1)) \right).
\end{aligned}$$

The right-hand side of this expression is clearly decreasing for large enough N, and in turn, finite. Hence, quadrature SCaSML boosts a quadrature MLP with complexity $O(d\varepsilon^{-(4+\delta)})$ to a corresponding physics-informed inference solver with complexity $O(de(\hat{u})^{4+\delta}\varepsilon^{-(4+\delta)})$ □

This theorem provides a comprehensive characterization of the computational complexity of our method. The inclusion of the surrogate model error measure $e(\hat{u})$ in the complexity bound demonstrates how the quality of the surrogate model directly influences the computational efficiency of our approach. Specifically, a more accurate surrogate model (smaller $e(\hat{u})$) leads to a lower computational cost for achieving a given level of accuracy.

F. Proof for full-history Multilevel Picard Iteration

This section examines the theoretical guarantees of SCaSML employing full-history Multilevel Picard Iteration to address the `law of defects`. For simplicity, we consider the case where $\mu = 0$ and $\sigma = s\mathbf{I}_d (s \in \mathbb{R})$ in the proof. One need to select a density ρ satisfying $\mathbb{P}(\mathfrak{r}^{(l,i)} \leq b) = \int_0^b \rho(s)\, ds = b^{1-\alpha}, b \in (0,1), \alpha > 0$ for full-history MLP (Definition 2). In the numerical experiments, we select $\rho(s) = (1-\alpha)s^{-\alpha}$.

F.1. Assumptions of the Surrogate Model

Relative to the quadrature discretization scheme, the Monte Carlo scheme in the context of full-history MLP imposes less stringent requirements on the surrogate model to attain a product convergence rate. In this section, we explicate the assumptions inherent in the error measure $e(\hat{u})$—a metric that evaluates the accuracy of the surrogate model and supports the improved convergence rates realized by the SCaSML methodology.

Assumption 1 (Accuracy of the Surrogate Model for full-history MLP). *Let* $\sup_{t\in[0,T]} \|\breve{u}^\infty(t,\cdot)\|_{W^{1,\infty}} < \infty$. *There exist constants* $C_{F,1}, C_{F,2} > 0$ *such that:*
1. L^∞ *Residual Bound:* $\sup_{r\in[0,T],y\in\mathbb{R}^d} |\varepsilon_{PDE}(r,y)| \leqslant C_{F,1}\, e(\hat{u})$,

2. $W^{1,\infty}$ *Error Bound:* $\sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{F,2}\, e(\hat{u})$, Note that the L^∞ operator bound, which comes from the time discretization error of the quadrature rule, in assumption 1 is avoided, due to the unbiasedness of the Monte Carlo time integral. This elimination significantly reduced the regularity requirement on the solution, indicating better robustness of the Monte Carlo method for malformed situations.

Assumption 2 carries over to the full-history setting in a weaker form. Similarly, the following integrability assumption from literature is needed (Hutzenthaler et al., 2021, Lemma 3.3):

Assumption 2 (full-history Integrability Assumption)**.** *There exists $p \in \mathbb{N}$ such that for the zero function initializer $\mathbf{0}$ and for all $t \in [0, T)$ and $q \in [1, p)$,*

$$\int_0^1 \frac{1}{s^{q/2}\rho(s)^{q-1}}\, ds + \sup_{s\in[t,T)} \mathbb{E}\left[\left|\breve{F}(\mathbf{0})(t,\, x + \sigma W_s - \sigma W_t)\right|^q\right] < \infty. \tag{65}$$

Contrary to the integrability assumption 2 in the quadrature case, the Monte Carlo scheme shifts the requirement on the terminal term to the time distribution from which we sample steps. Intuitively, it requires a larger α to allocate more probability near time 0, where the information from u is relatively scarce, compared to the terminal given at time T. Also, the integrability condition on the nonlinearity is strengthened to remain integrable under perturbation to deal with less regular situation.

F.2. Main Results

We now show that, with an appropriately trained surrogate model, the `Law of defect` can be simulated with lower complexity than the original PDE. In particular, the error of the full-history MLP is upper-bounded by the surrogate model's error measure $e(\hat{u})$.

F.2.1. Bound on Global L^2 Error

Our proof still utilizes the insight that the overall L^2 error in the MLP mainly hinges on the Lipschitz continuity of the PDE's terminal and solution, as well as the extent of nonlinearity at the origin. We illustrate that the parameter linked to the `Law of defect` is constrained by the surrogate error. Initially, we present a lemma demonstrating how the complexity of MLP can be capped by the error assessment.

Lemma 3 (Complexity Estimation via Surrogate Model's Error Measure $e(\hat{u})$)**.** *Under assumptions 5, 6, 2 and 1 , suppose $p \geqslant 2$, $x \in \mathbb{R}^d$. There exists a constant $C_F > 0$ such that for all $M, N \geq 2$,*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \left\{ \frac{\sigma\sqrt{\max\{T-t, 3\}}\|\breve{K}\|_1}{\sqrt{M}} + \frac{C\ \sup_{s\in[t,T)} \|\breve{F}(\mathbf{0})(s, x+\sigma W_s - \sigma W_t)\|_{L^{\frac{2p}{p-2}}}}{2\sqrt{M}} \right. \tag{66}$$

$$\left. + \frac{C\ \sup_{s\in[t,T),\nu\in\{1,\ldots,d+1\}} \|\breve{\mathbf{u}}(s, x+\sigma W_s - \sigma W_t)_\nu\|_{L^{\frac{2p}{p-2}}}}{2} \right\} \leq C_F\, e(\hat{u}), \tag{67}$$

where

$$C = \max\left\{1, 2T^{\frac{1}{2}}|\Gamma(\frac{p}{2})|^{\frac{1}{p}}(1-\alpha)^{\frac{1}{p}-1}\max\{1, \|\breve{L}\|_1\}\max\left\{T^{\frac{1}{2}}, 2^{\frac{1}{2}}|\Gamma(\frac{p+1}{2})|^{\frac{1}{p}}\pi^{-\frac{1}{2p}}\right\}\right\}. \tag{68}$$

Proof. The proof is almost identical with 3. We use the accuracy of the surrogate model 1 to control each term.

Suppose e_α is the unit vector with 1 at its α-th component. For the first term, we note that $\breve{K}$ is the vector of Lipschitz constants of $\breve{g}$, so it must holds that $\breve{K}_\alpha \leqslant \|D^{e_\alpha}\breve{g}\|_{L^\infty}$(Just check the definition in 7) $\Rightarrow$ $\|\breve{K}\|_1 = \sum_{\alpha=1}^{d} \breve{K}_\alpha \leqslant \sum_{\alpha=1}^{d} \|D^{e_\alpha}\breve{g}\|_{L^\infty} = \sum_{\alpha=2}^{d+1} \|D^{e_\alpha}u(T,\cdot)\|_{L^\infty}$, and the RHS can be controlled by $W^{1,\infty}$ error bound in assumption 1. Combing this fact and definition of Sobolev norm 3, we get:

$$\|\breve{K}\|_1 \leqslant \sum_{\alpha=1}^{d} \|D^{e_\alpha}\breve{g}\|_{L^\infty} \leqslant \sum_{\alpha=2}^{d+1} \|D^{e_\alpha}\breve{u}^\infty(T,\cdot)\|_{L^\infty} + \|\breve{u}^\infty(T,\cdot)\|_{L^\infty} = \|\breve{u}(T,\cdot)\|_{W^{1,\infty}} \tag{69}$$

$$\leqslant \sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{F,2}e(\hat{u}). \tag{70}$$

For the second term, recalling that $\breve{F}(\mathbf{v}) := F((\hat{u},\sigma\nabla_y\hat{u})+\mathbf{v}) - F((\hat{u},\sigma\nabla_y\hat{u})) + \varepsilon_{PDE} \Rightarrow \breve{F}(\mathbf{0}) = \varepsilon_{PDE}$ and that under probability measure we have $\|\cdot\|_{L^{\frac{2p}{p-2}}} \leqslant \|\cdot\|_{L^\infty}$, we apply L^∞ residual bound in assumption 1:

$$\sup_{s\in[t,T)} \left\|\breve{F}(\mathbf{0})(s,x+\sigma W_s-\sigma W_t)\right\|_{L^{\frac{2p}{p-2}}} = \sup_{s\in[t,T)} \|(\varepsilon_{\text{PDE}})(s,x+\sigma W_s-\sigma W_t)\|_{L^{\frac{2p}{p-2}}} \tag{71}$$

$$\leqslant \sup_{r\in[0,T],y\in\mathbb{R}^d} |\varepsilon_{\text{PDE}}(r,y)| \leqslant C_{F,1}e(\hat{u}) \tag{72}$$

For the third term, recalling that $\breve{\mathbf{u}} = (\breve{u}^\infty, \sigma\nabla_y\breve{u}^\infty)$ and that under probability measure we have $\|\cdot\|_{L^{\frac{2p}{p-2}}} \leqslant \|\cdot\|_{L^\infty}$, we apply the $W^{1,\infty}$ error bound in assumption 1 again:

$$\sup_{s\in[t,T),\nu\in\{1,2,\dots,d+1\}} \|\breve{\mathbf{u}}(s,x+\sigma W_s-\sigma W_t)_\nu\|_{L^{\frac{2p}{p-2}}} \leqslant \sup_{s\in[t,T)} \max_{\nu\in\{1,2,\dots,d+1\}} \|\breve{\mathbf{u}}(s,x)_\nu\|_{L^\infty} \tag{73}$$

$$\leqslant \sum_{\alpha=2}^{d+1} \|D^{e_\alpha}\breve{u}^\infty(T,\cdot)\|_{L^\infty} + \|\breve{u}^\infty(T,\cdot)\|_{L^\infty} \leqslant \|\breve{u}(T,\cdot)\|_{W^{1,\infty}} \tag{74}$$

$$\leqslant \sup_{r\in[0,T]} \|\breve{u}(r,\cdot)\|_{W^{1,\infty}} \leqslant C_{F,2}e(\hat{u}). \tag{75}$$

Plugging 69, 71 and 73 in LHS of 3, and define $C_F = \frac{C}{2}C_{F,1} + (\sigma\sqrt{\max\{T-t,3\}} + \frac{C}{2})C_{F,2}$, we have

$$\frac{\sigma\sqrt{\max\{T-t,3\}}\|\breve{K}\|_1}{\sqrt{M}} + \frac{C\ \sup_{s\in[t,T)} \|\ \breve{F}(\mathbf{0})(s,x+\sigma W_s-\sigma W_t)\|_{L^{\frac{2p}{p-2}}}}{2\sqrt{M}} \tag{76}$$

$$+\ \frac{C\ \sup_{s\in[t,T),\nu\in\{1,\dots,d+1\}} \|\breve{\mathbf{u}}(s,x+\sigma W_s-\sigma W_t)_\nu\|_{L^{\frac{2p}{p-2}}}}{2} \tag{77}$$

$$\leqslant \left(\frac{C}{2}C_{F,1} + (\sigma\sqrt{\max\{T,3\}} + \frac{C}{2})C_{F,2}\right)e(\hat{u}) \leqslant C_F\, e(\hat{u}). \tag{78}$$

Noting that the RHS is independent of (t,x), we take $\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}$ for both side, and proved the lemma. □

The above lemma, together with standard error estimates for the full-history MLP, yields the following result.

Theorem 4 (Bound of Global L^2 Error). *Under assumptions 5, 6, 2 and 1 , suppose $p \geqslant 2$, $\alpha \in (\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)})$, $t \in [0,T)$, $x \in \mathbb{R}^d$, $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$. It holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\breve{\mathbf{U}}_{N,M}(t,x) - \breve{\mathbf{u}}(t,x)\right)_\nu\right\|_{L^2} \leq E(M,N)\cdot\left(C_F\, e(\hat{u})\right), \tag{79}$$

where

$$E(M,N) = \frac{\left[e\left(\frac{pN}{2}+1\right)\right]^{\frac{1}{8}}(2C)^{N-1}\exp\left(\beta M^{\frac{1}{2\beta}}\right)}{\sqrt{M^{N-1}}}. \tag{80}$$

Proof. Under assumptions 2 and 7, combined with integrability argument in (Hutzenthaler et al., 2021, Lemma 3.3), the proof of (Hutzenthaler et al., 2021, Proposition 3.5) holds. Setting $n = N$ in this proposition, for all $\nu \in \{1,\dots,d+1\}$, we have

$$\begin{aligned}
\left\|\left(\breve{\mathbf{U}}_{N,M}(t,x)-\breve{\mathbf{u}}(t,x)\right)_\nu\right\|_{L^2} \leq & E(M,N)\cdot\left\{\frac{\sigma\sqrt{\max\{T-t,3\}}\|\breve{K}\|_1}{\sqrt{M}}\right. & (81)\\
& + \frac{C\,\sup_{s\in[t,T)}\|\breve{F}(\mathbf{0})(s,x+\sigma W_s-\sigma W_t)\|_{L^{\frac{2p}{p-2}}}}{2\sqrt{M}} & (82)\\
& \left.+ \frac{C\,\sup_{s\in[t,T),\nu\in\{1,\dots,d+1\}}\|\breve{\mathbf{u}}(s,x+\sigma W_s-\sigma W_t)_\nu\|_{L^{\frac{2p}{p-2}}}}{2}\right\}. & (83)
\end{aligned}$$

Take $\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\max_{\nu\in\{1,\dots,d+1\}}$ for the LHS, and note that the RHS does not depend on ν, we get

$$\begin{aligned}
& \sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\ \max_{\nu\in\{1,\dots,d+1\}}\left\|\left(\breve{\mathbf{U}}_{N,M}(t,x)-\breve{\mathbf{u}}(t,x)\right)_\nu\right\|_{L^2} & (84)\\
\leq & E(M,N)\cdot\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\left\{\frac{\sigma\sqrt{\max\{T-t,3\}}\|\breve{K}\|_1}{\sqrt{M}}\right. & (85)\\
& + \frac{C\,\sup_{s\in[t,T)}\|\breve{F}(\mathbf{0})(s,x+\sigma W_s-\sigma W_t)\|_{L^{\frac{2p}{p-2}}}}{2\sqrt{M}} & (86)\\
& \left.+ \frac{C\,\sup_{s\in[t,T),\nu\in\{1,\dots,d+1\}}\|\breve{\mathbf{u}}(s,x+\sigma W_s-\sigma W_t)_\nu\|_{L^{\frac{2p}{p-2}}}}{2}\right\}. & (87)
\end{aligned}$$

Substituting the sup term in 84 by Lemma 3 immediately yields the stated result. □

In practice, a common choice for M is $\lfloor N^{2\beta}\rfloor$. Plugging it in 5, we get the error order of the solver w.r.t. N:

Corollary 5 (Error Order for $M = \lfloor N^{2\beta}\rfloor$)**.** *Under assumptions 5, 6, 2 and 1 , suppose* $p \geqslant 2$, $\alpha \in (\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)})$, $t \in [0,T)$, $x \in \mathbb{R}^d$, $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$. *It holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\ \max_{\nu\in\{1,\dots,d+1\}}\left\|\left(\breve{\mathbf{U}}_{N,M}(t,x)-\breve{\mathbf{u}}(t,x)\right)_\nu\right\|_{L^2} \leq \exp\Big(N\log N(-\beta+o(1))\Big)\cdot\Big(C_F\,e(\hat{u})\Big). \tag{88}$$

Proof. First, we rewrite $E(M,N)$ in 4 as exponential form:

$$\begin{aligned}
E(M,N) = & \frac{\left[e\left(\frac{pN}{2}+1\right)\right]^{\frac{1}{8}}(2C)^{N-1}\exp\left(\beta M^{\frac{1}{2\beta}}\right)}{\sqrt{M^{N-1}}} & (89)\\
= & \exp\left(o(N)+N\log(2C)+\beta M^{1/2\beta}-\frac{N-1}{2}\log M\right) & (90)
\end{aligned}$$

Note that $\lfloor N^{2\beta} \rfloor \leqslant N^{2\beta}$ and that $M = \lfloor N^{2\beta} \rfloor \Rightarrow \log M \geqslant \log(N^{2\beta} - 1) \geqslant \log(N^{2\beta}) - 1 = 2\beta \log N - 1$ for $N \geqslant 2^{1/2\beta}$. We can simplify 89 to

$$\exp\left(o(N) + N\log(2C) + \beta M^{1/2\beta} - \frac{N-1}{2}\log M\right) \tag{91}$$

$$\leqslant \exp\left(o(N) + N\log(2C) + \beta N - \frac{N-1}{2}(2\beta\log N - 1)\right) \tag{92}$$

$$= \exp\left(N(o(1) + \log(2C) + \beta - \frac{1}{2}(2\beta\log N - 1))\right) \tag{93}$$

$$= \exp\left(N\log N(-\beta + o(1))\right). \tag{94}$$

Plugging 91 to the conclusion of 4, we get the result we want. □

Corollary 6 (Improved Scaling Law for $M = \lfloor N^{2\beta} \rfloor$). *Under Assumptions 5, 6, 2 and 1, suppose that* $p \geqslant 2$, $\alpha \in \left(\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)}\right)$, $t \in [0,T)$, $x \in \mathbb{R}^d$, *and define* $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$. *Assume that the error at* (t,x) *of the surrogate model decays polynomially with respect to the number of training points; namely,* $e(\hat{u}) = O(m^{-\gamma})$, *for some* $\gamma > 0$. *Suppose further that* $m = (d+1)5^N N^{2\beta}$. *Then, for all sufficiently large* m, *the SCaSML procedure improves the error bound from* $O(m^{-\gamma})$ *to* $O\left(m^{-\gamma-\frac{1}{2}+o(1)}\right)$ *with same points number.*

Proof. In what follows, we adopt the notation $f(m) \sim g(m)$ to signify that $\lim_{m\to\infty} \frac{f(m)}{g(m)} = 1$. Since m is a continuous and strictly increasing function of N, there exists a unique inverse function $N = N(m)$. Taking logarithms, we obtain $\log m = \log(d+1) + N(m)\log 5 + 2\beta\, N(m)\log N(m)$ which follows immediately that $\log m \sim 2\beta\, N(m)\log N(m)$

Define

$$z = \frac{\log m}{2\beta} - \frac{\log(d+1) + N(m)\log 5}{2\beta}$$

and set $x = \log N$, so that the relation $x\,e^x = z$ holds. The inverse of this equation is given by the Lambert W function, i.e., $x = W(z)$. Therefore,

$$N(m) = e^x = e^{W(z)} = \frac{z}{W(z)} = \frac{\frac{\log m}{2\beta} - \frac{\log(d+1)+N(m)\log 5}{2\beta}}{W\left(\frac{\log m}{2\beta} - \frac{\log(d+1)+N(m)\log 5}{2\beta}\right)}.$$

Since $W(z) \sim \log z - \log\log z$, we can deduce that $N(m) \sim \frac{\frac{\log m}{2\beta}}{\log(\log m)} = \frac{\log m}{2\beta\, \log\log m}$. Equivalently, $N(m) = \frac{\log m}{2\beta\, \log\log m} + o\left(\frac{\log m}{2\beta\, \log\log m}\right)$.

In contrast to the surrogate model, which uses all m points to achieve an error of $O(m^{-\gamma})$, the SCaSML method allocates $5^N N^{2\beta}$ points for training and $d5^N N^{2\beta}$ points for inference (see Footnote 7), thereby yielding an error bound of the form $O\left(N\log N\,(-\beta + o(1))\,(5^N N^{2\beta})^{-\gamma}\right) =$

$O\Big(N^{-\beta N(1+o(1))}\, m^{-\gamma}\Big)$. Substituting the asymptotic expression for $N(m)$, and noting that

$$\begin{aligned}
N(m)^{-\beta N(m)} =& \frac{\sqrt{d+1}\exp(\frac{\log 5}{2}N(m))}{\sqrt{m}} \\
=& \sqrt{d+1}\exp(\frac{\log 5}{2}\log m(\frac{N(m)}{\log m}-\frac{1}{\log 5})) \\
=& \sqrt{d+1}\exp(\frac{\log 5}{2}\log m(-\frac{1}{\log 5}+o(\frac{1}{\log\log m}))) \\
=& \sqrt{d+1}\exp(-(\frac{1}{2}-o(\frac{1}{\log\log m}))\log m) \\
=& \sqrt{d+1}m^{-\frac{1}{2}+o(\frac{1}{\log\log m})} = O(m^{-\frac{1}{2}+o(\frac{1}{\log\log m})}).
\end{aligned}$$

We obtain the SCaSML error bound $O\Big(m^{-\gamma}\, m^{(-\frac{1}{2}+o(\frac{1}{\log\log m}))(1+o(1))}\Big) = O\Big(m^{-\gamma-\frac{1}{2}+o(1)}\Big)$. Hence, for high-dimensional problems where $m \gg 1$ and for any fixed $\gamma > 0$, we conclude that $O\Big(m^{-\gamma-\frac{1}{2}+o(1)}\Big) \ll O\Big(m^{-\gamma}\Big)$, thereby demonstrating that the SCaSML procedure attains a strictly faster rate of convergence. □

Corollary 7 (Error Order for $M = \lfloor N^{2\beta}\rfloor$)**.** *Under Assumptions 1, 2, 5 and 6 , suppose $p \geqslant 2$, $\alpha \in (\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)})$, $t \in [0,T)$, $x \in \mathbb{R}^d$, $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p}$. It holds that*

$$\sup_{(t,x)\in[0,T]\times\mathbb{R}^d}\ \max_{\nu\in\{1,\ldots,d+1\}} \left\| \Big(\breve{\mathbf{U}}_{N,\lfloor N^{2\beta}\rfloor}(t,x) - \breve{\mathbf{u}}(t,x)\Big)_\nu \right\|_{L^2} \le \exp\Big(N\log N(-\beta+o(1))\Big)\cdot\Big(C_F\, e(\hat{u})\Big). \tag{95}$$

Specifically, this approximator $\breve{\mathbf{U}}_{N,\lfloor N^{2\beta}\rfloor}$ requires at most $d(5\lfloor N^{2\beta}\rfloor)^N$ points for evaluation, as detailed in (Hutzenthaler et al., 2020a, Lemma 3.6).

F.2.2. Bound on Computational Complexity

We now define two indicators to quantify the computational complexity of full-history SCaSML:the number of realization variables (RV) and the number of function evaluations (FE).

Definition 8 (Computational Complexity of full-history SCaSML)**.** *We define the following complexity:*

- *Let $\{\mathrm{RV}_{n,M}\}_{n,M\in\mathbb{Z}} \subset \mathbb{N}$ satisfy $\mathrm{RV}_{0,M} = 0$ and, for all $n, M \in \mathbb{N}$,*

$$\mathrm{RV}_{n,M} \le d\,M^n + \sum_{l=0}^{n-1}\Big[M^{n-l}\Big(1+d+\mathrm{RV}_{l,M} + \mathbf{1}_{\mathbb{N}}(l)\ \mathrm{RV}_{l-1,M}\Big)\Big]. \tag{96}$$

  This quantity captures the number of scalar normal and uniform time realizations required to compute one sample of $\breve{\mathbf{U}}_{n,M}(s,x)$.
- *Let $\{\mathrm{FE}_{n,M}\}_{n,M\in\mathbb{Z}} \subset \mathbb{N}$ satisfy $\mathrm{FE}_{0,M} = 0$ and, for all $n, M \in \mathbb{N}$,*

$$\mathrm{FE}_{n,M} \le M^n + \sum_{l=0}^{n-1}\Big[M^{n-l}\Big(1+\mathrm{FE}_{l,M} + \mathbf{1}_{\mathbb{N}}(l) + \mathbf{1}_{\mathbb{N}}(l)\ \mathrm{FE}_{l-1,M}\Big)\Big]. \tag{97}$$

  This reflects the number of evaluations of $\breve{F}$ and $\breve{g}$ required to compute one sample of $\breve{\mathbf{U}}_{n,M}(s,x)$.

Theorem 9 (Computational Complexity of full-history SCaSML)**.** *Under assumptions 5, 6, 2 and 1 , suppose $p \geqslant 2$, $\alpha \in (\frac{p-2}{2(p-1)}, \frac{p}{2(p-1)})$ and $\beta = \frac{\alpha}{2} - \frac{(1-\alpha)(p-2)}{2p} \in (0, \frac{\alpha}{2})$. For any $N \geqslant 2$ and $\delta > 0$, taking $M = \lfloor N^{2\beta} \rfloor$, we have*

$$\mathrm{RV}_{N,M} + \mathrm{FE}_{N,M} \leqslant \exp\left(N \log N\Big(- \beta\delta + o(1)\Big)\right)$$
$$(d+1)(C_F e(\hat{u}))^{2+\delta}\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(2+\delta)}. \tag{98}$$

Proof. From (Hutzenthaler et al., 2020a, Lemma 3.6), we derive that

$$\mathrm{RV}_{N,M} \leqslant d(5M)^N, \mathrm{FE}_{N,M} \leqslant (5M)^N. \tag{99}$$

Suppose the maximum error is ε. To compensate for the $(5M)^N$ term in the complexity by the denominator of Theorem 4, we multiply the complexity by $\varepsilon^{2+\delta}$ and then divide it, and put everything into the exponent:

$$\mathrm{RV}_{N,M} + \mathrm{FE}_{N,M} \tag{100}$$
$$\leqslant (d+1)(5M)^N \tag{101}$$
$$= (d+1)(5M)^N \varepsilon^{2+\delta}\varepsilon^{-(2+\delta)} \tag{102}$$
$$\leqslant \left[\frac{\left[e\left(\frac{pN}{2}+1\right)\right]^{\frac{1}{8}} (2C)^{N-1} \exp\left(\beta M^{\frac{1}{2\beta}}\right)}{\sqrt{M^{N-1}}}\right]^{2+\delta} (C_F e(\hat{u}))^{2+\delta}(d+1)(5M)^N \tag{103}$$
$$\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(2+\delta)} \tag{104}$$
$$= \exp\left(N\Big((2+\delta)\log(2C) + \log 5\Big) + (2+\delta)\beta M^{1/2\beta} + \log M - \frac{\delta}{2}(N-1)\log M + o(N)\right) \tag{105}$$
$$(d+1)(C_F e(\hat{u}))^{2+\delta}\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(2+\delta)} \tag{106}$$
$$= \exp\left(N\Big((2+\delta)\log(2C) + \log 5 + \frac{(2+\delta)\beta}{N}M^{1/2\beta} - (\frac{\delta}{2} - \frac{2+\delta}{2N})\log M + o(1)\Big)\right) \tag{107}$$
$$(d+1)(C_F e(\hat{u}))^{2+\delta}\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(2+\delta)} \tag{108}$$
$$\leqslant \exp\left(N\Big((2+\delta)\log(2C) + \log 5 + \frac{(2+\delta)\beta}{N}M^{1/2\beta} - (\frac{\delta}{2} - \frac{2+\delta}{2N})\log M + o(1)\Big)\right) \tag{109}$$
$$(d+1)(C_F e(\hat{u}))^{2+\delta}\left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\dots,d+1\}} \left\|\left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu\right\|_{L^2}\right]^{-(2+\delta)} \tag{110}$$

Note that $\lfloor N^{2\beta} \rfloor^{1/2\beta} \leqslant N$ and $\beta < \frac{\alpha}{2}$, thus $\frac{(2+\delta)\beta}{N} M^{1/2\beta} \leqslant (2+\delta)\beta \leqslant \alpha(1+\frac{\delta}{2})$. Therefore, by 100:

$$\exp\left(N\left((2+\delta)\log(2C) + \log 5 + \frac{(2+\delta)\beta}{N} M^{1/2\beta} - (\frac{\delta}{2} - \frac{2+\delta}{2N}) \log M + o(1)\right)\right) \tag{111}$$

$$(d+1)(C_F e(\hat{u}))^{2+\delta} \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu \right\|_{L^2}\right]^{-(2+\delta)} \tag{112}$$

$$\leqslant \exp\left(N\left((2+\delta)\log(2C) + \log 5 + \alpha(1+\frac{\delta}{2}) - (\frac{\delta}{2} - \frac{2+\delta}{2N}) \log M + o(1)\right)\right) \tag{113}$$

$$(d+1)(C_F e(\hat{u}))^{2+\delta} \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu \right\|_{L^2}\right]^{-(2+\delta)}. \tag{114}$$

Since $M = \lfloor N^{2\beta} \rfloor \Rightarrow \log M \geqslant \log(N^{2\beta} - 1) \geqslant \log(N^{2\beta}) - 1 = 2\beta \log N - 1$ for $N \geqslant 2^{1/2\beta}$, we can further reduce 111 to:

$$\exp\left(N\left((2+\delta)\log(2C) + \log 5 + \alpha(1+\frac{\delta}{2}) - (\frac{\delta}{2} - \frac{2+\delta}{2N}) \log M + o(1)\right)\right) \tag{115}$$

$$(d+1)(C_F e(\hat{u}))^{2+\delta} \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu \right\|_{L^2}\right]^{-(2+\delta)} \tag{116}$$

$$\leqslant \exp\left(N\left((2+\delta)\log(2C) + \log 5 + \alpha(1+\frac{\delta}{2}) + (\frac{\delta}{2} - \frac{2+\delta}{2N}) - (\frac{\delta}{2} - \frac{2+\delta}{2N}) \cdot 2\beta \log N + o(1)\right)\right) \tag{117}$$

$$(d+1)(C_F e(\hat{u}))^{2+\delta} \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu \right\|_{L^2}\right]^{-(2+\delta)} \tag{118}$$

$$= \exp\left(N \log N\left(-\beta\delta + o(1)\right)\right) \tag{119}$$

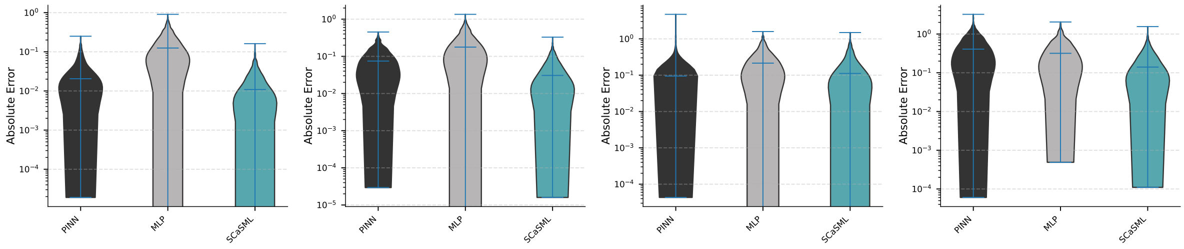
$$(d+1)(C_F e(\hat{u}))^{2+\delta} \left[\sup_{(t,x)\in[0,T]\times\mathbb{R}^d} \max_{\nu\in\{1,\ldots,d+1\}} \left\| \left(\tilde{\mathbf{U}}_{N,M}(t,x) - \mathbf{u}^\infty(t,x)\right)_\nu \right\|_{L^2}\right]^{-(2+\delta)}. \tag{120}$$

The right-hand side of this expression is clearly decreasing for large enough N, and in turn, finite. Hence, quadrature SCaSML boosts a quadrature MLP with complexity $O(d\varepsilon^{-(2+\delta)})$ to a corresponding physics-informed inference solver with complexity $O(de(\hat{u})^{2+\delta}\varepsilon^{-(2+\delta)})$. □

G. Auxiliary Experiments results

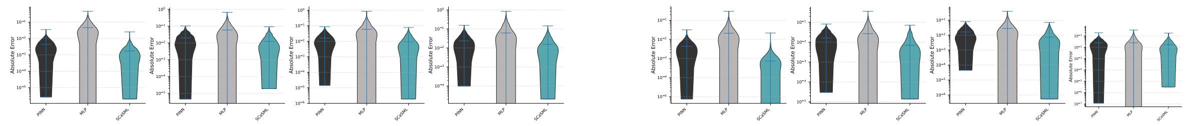
G.1. Violin Plot for Error distribution

In this section, we present violin plots of the absolute error distributions for the base surrogate model , the MLP, and the SCaSML method. We uniformly select the test points. By combining kernel density estimation with boxplot-style summaries, these plots capture both the spread and central tendency of the errors. A violin plot exposes the full distribution—its density, variability, skewness, and outliers—offering much deeper insight into model performance. The width of each violin at a given error level reflects the density of the observations. The results indicate that SCaSML reduces the largest absolute error, lowers the median and produces more accurate points for a majority of equations compared to the surrogate and MLP, demonstrating its robustness across different dimensions and equations.



(a) $d = 10$ (b) $d = 20$ (c) $d = 30$ (d) $d = 60$

Figure 4 | Violin Plot for comparison of the baseline PINN surrogate (black), MLP (gray), applying qudrature SCaSML (teal) to calibrate the PINN surrogate on linear convection-diffusion equation for $d = 10, 20, 30, 60$.



$d = 20$ $d = 40$ $d = 60$ $d = 80$ $d = 20$ $d = 40$ $d = 60$ $d = 80$

(a) SCaSML using Quadrature MLP (b) SCaSML using full-history MLP

Figure 5 | Violin Plot for comparison of the baseline PINN surrogate (black), MLP (gray), applying qudrature SCaSML (teal) to calibrate the PINN surrogate on viscous Burgers' equation equation for $d = 10, 20, 30, 60$.

G.2. Inference Time Scaling Curve

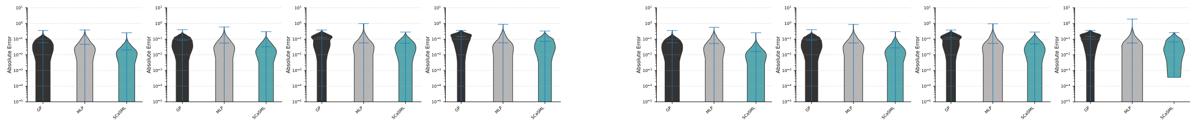
In this section, we illustrate how SCaSML enhances estimation accuracy as the number of inference-time collocation points increases, as outlined in 2.1 and 3.2. Our findings indicate that allocating additional computational resources during inference consistently improves estimation accuracy.

G.3. Improved Scaling Law of SCaSML Algorithms

In this section, we consider the viscous Burgers equation as an illustrative example to demonstrate the improved convergence of SCaSML algorithms, as suggested by Corollary 6.

We implemented a physics-informed neural network (PINN) with five hidden layers, each containing 50 neurons and employing hyperbolic tangent activation functions. Because the number of training points, m, is proportional to the number of iterations in the PINN, the control group was trained using the Adam optimizer (learning rate 7×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.99$) over iterations set to 400, 2 000, 4 000, 6 000, 8 000, and 10 000 (as illustrated along the x-axis). The dataset comprised 2 500 interior points, 100 boundary points, and 160 initial points uniformly sampled from $[0, 0.5] \times [-0.5, 0.5]^d$, ensuring that $m \gg 1$. To replicate the conditions of Corollary 6, the SCaSML group was trained over iterations set to $\lfloor 400/(d+1) \rfloor$, $\lfloor 2\,000/(d+1) \rfloor$, $\lfloor 1\,000/(d+1) \rfloor$, $\lfloor 6\,000/(d+1) \rfloor$, $\lfloor 8\,000/(d+1) \rfloor$, and $\lfloor 10\,000/(d+1) \rfloor$. In addition, we set the inference level as $N = \lfloor \log m/(2\beta \log\log m) \rfloor$ with $\beta = 1/2$. Theoretically, SCaSML exhibits an improvement in γ of $\frac{1}{2} + o(1)$ relative to the control group.

For the Gaussian process regression surrogate model, training was performed over 20 iterations using Newton's method. Due to the increasing inference parameters with m and the consequent GPU memory constraints, it was not possible to replicate the conditions of Corollary



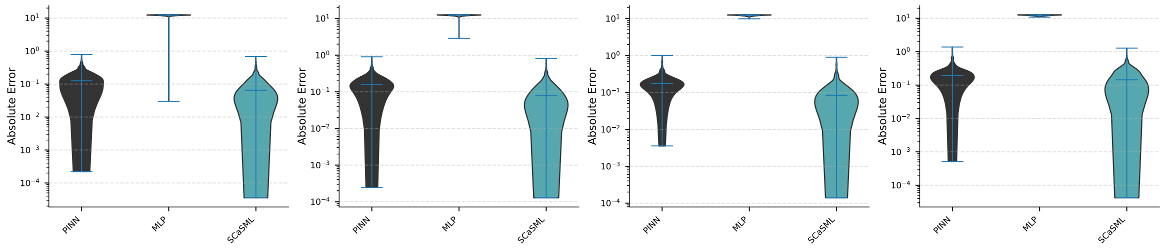
$d = 20$ $d = 40$ $d = 60$ $d = 80$

(a) SCaSML using Quadrature MLP

$d = 20$ $d = 40$ $d = 60$ $d = 80$

(b) SCaSML using full-history MLP

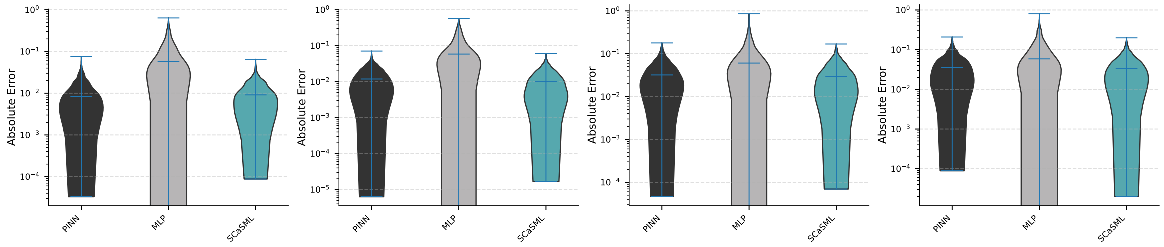
Figure 6 | Violin Plot for comparison of the baseline Gaussian Process surrogate (black), MLP (gray), applying qudrature SCaSML (teal) to calibrate the Gaussian Process surrogate on viscous Burgers' equation equation for $d = 20, 40, 60, 80$.



$d = 100$ $d = 120$ $d = 140$ $d = 160$

(a) SCaSML using full-history MLP

Figure 7 | Violin Plot for comparison of the baseline PINN surrogate (black), MLP (gray), applying qudrature SCaSML (teal) to calibrate the PINN surrogate on LQG control problem for $d = 100, 120, 140, 160$.



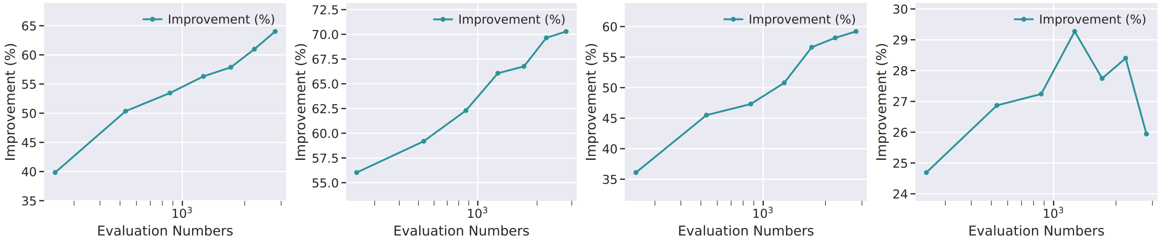
$d = 100$ $d = 120$ $d = 140$ $d = 160$

(a) SCaSML using full-history MLP

Figure 8 | Violin Plot for comparison of the baseline PINN surrogate (black), MLP (gray), applying qudrature SCaSML (teal) to calibrate the PINN surrogate on diffusion reaction equation for $d = 100, 120, 140, 160$.

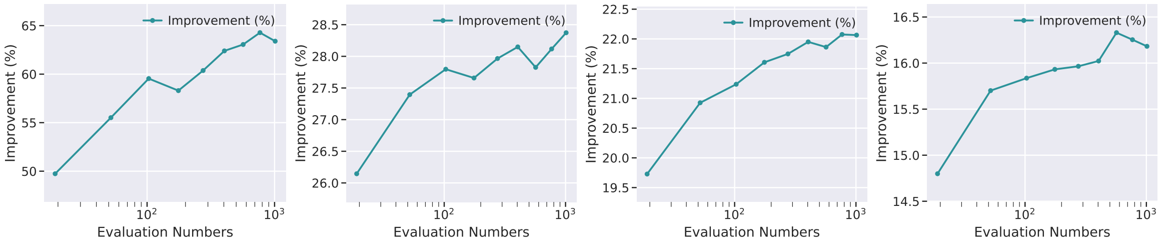
6 exactly for the Gaussian process model. Consequently, both the control and SCaSML groups employed identical training sizes, which theoretically does not alter the asymptopic convergence rate(i.e. the slope). Specifically, the training data consisted of the following pairs of interior and boundary points:(100, 20), (200, 40), (300, 60), (400, 80), (500, 100), (600, 120), (700, 140), (800, 160), (900, 180), and (1 000, 200), with the x-axis representing the total number of training points. Again, the inference level was chosen as $N = \lfloor \log m/(2\beta \log\log m) \rfloor$ with $\beta = 1/2$, and the SCaSML continues to exhibit an improvement in γ of $\frac{1}{2} + o(1)$ relative to the control group.

We observe that, for the PINNs, full-history SCaSML achieves near-monotonic error reduction across resolutions (with d ranging from 20 to 80), outperforming quadrature SCaSML,



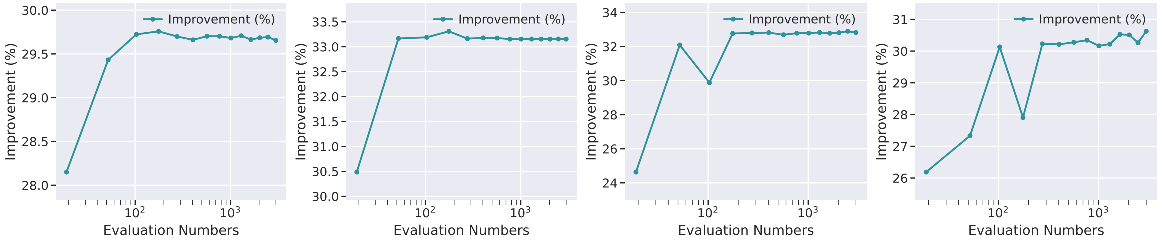
(a) $d = 10$ (b) $d = 20$ (c) $d = 30$ (d) $d = 60$

Figure 9 | For the linear convection-diffusion equation, SCaSML for PINNs reliably enhances performance with increased computational resources. Notably, scaling effects are more pronounced in lower dimensions, potentially due to the MLP's convergence rate exhibiting a linear dependency on the dimensionality d.



(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

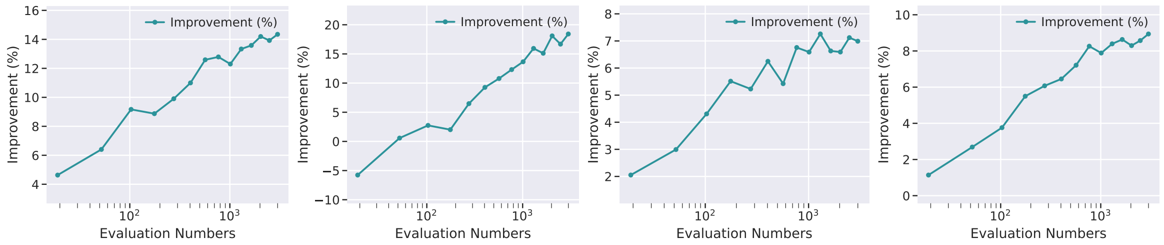
Figure 10 | For the viscous Burgers equation, SCaSML with PINN consistently improves performance as the sample size M increases exponentially.



(a) $d = 100$ (b) $d = 120$ (c) $d = 140$ (d) $d = 160$

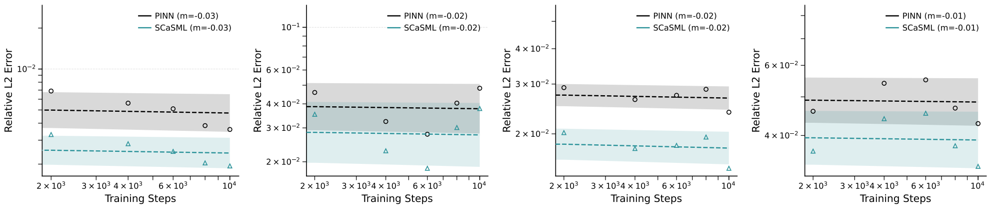
Figure 11 | For the HJB equation, SCaSML with PINN consistently enhances performance with increases in the exponential base of the sample size M. However, the scaling curve plateaus at $M = 14$, likely due to the relatively small clipping range of SCaSML compared to the solution magnitude. In general, a larger clipping threshold permits more outliers, thereby requiring additional samples to mitigate variance and ultimately enhancing accuracy; this trade-off must be considered in light of available computational resources.

which displays oscillatory behavior at higher dimensions. The Gaussian process-based SCaSML similarly accelerates convergence during training. In both cases, the error trajectories generated by SCaSML are generally shifted downward relative to the base models, underscoring its capacity to enhance accuracy without altering the fundamental training dynamics. These findings underscore SCaSML's robustness in diverse settings, ensuring reliable convergence even in high-dimensional or non-monotonic scenarios.



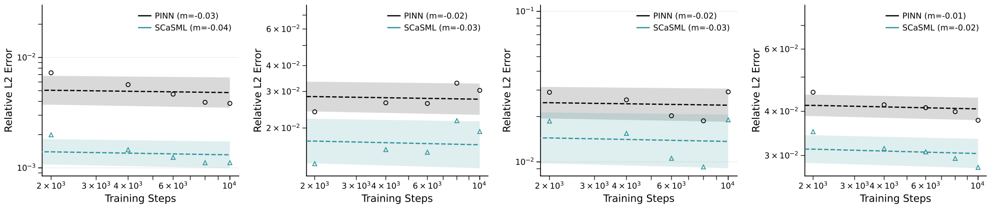
(a) $d = 100$ (b) $d = 120$ (c) $d = 140$ (d) $d = 160$

Figure 12 | For the Diffusion Reaction equation, SCaSML with PINN consistently improves performance as the exponential base of the sample size M increases.



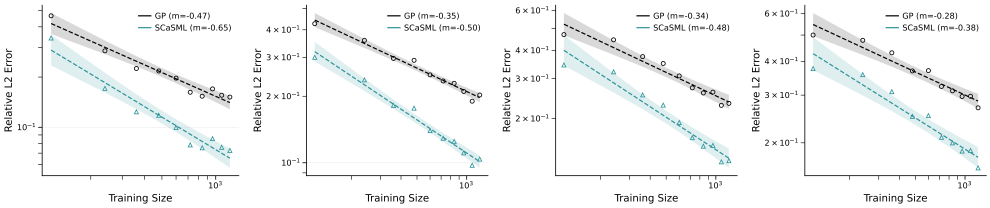
(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

Figure 13 | We apply qudrature SCaSML to calibrate a PINN surrogate for the d-dimensional viscous Burgers equation. All plots employ logarithmic scales on both axes, and the slope γ denotes the polynomial convergence rate. Numerical results demonstrate that, when collocation points for testing and inference are increased simultaneously, SCaSML achieves a faster scaling law than the base surrogate model.



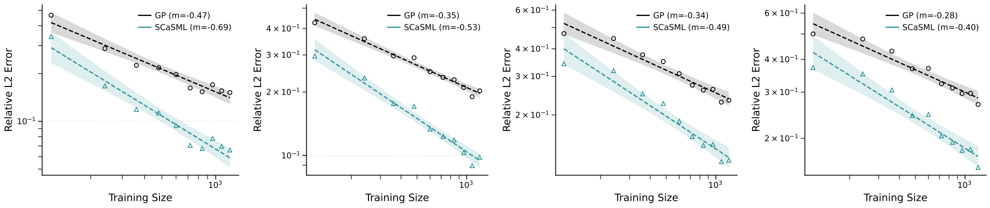
(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

Figure 14 | We apply full-history SCaSML to calibrate a PINN surrogate for the d-dimensional viscous Burgers equation. All plots employ logarithmic scales on both axes, and the slope γ denotes the polynomial convergence rate. Numerical results demonstrate that, when collocation points for testing and inference are increased simultaneously, SCaSML achieves a faster scaling law than the base surrogate model.



(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

Figure 15 | We apply quadrature SCaSML to calibrate a Gaussian Process surrogate for the d-dimensional viscous Burgers equation. All plots employ logarithmic scales on both axes, and the slope γ denotes the polynomial convergence rate. Numerical results demonstrate that, when collocation points for testing and inference are increased simultaneously, SCaSML achieves a faster scaling law than the base surrogate model.



(a) $d = 20$ (b) $d = 40$ (c) $d = 60$ (d) $d = 80$

Figure 16 | We apply full-history SCaSML to calibrate a Gaussian Process surrogate for the d-dimensional viscous Burgers equation. All plots employ logarithmic scales on both axes, and the slope γ denotes the polynomial convergence rate. Numerical results demonstrate that, when collocation points for testing and inference are increased simultaneously, SCaSML achieves a faster scaling law than the base surrogate model.