# Solving PDEs via deep learning

## SDE-based variational formulations

Julius Berner

October 17, 2022

University of Vienna

# Table of contents

# Introduction to numerical methods for PDEs

# Partial differential equation (PDEs)

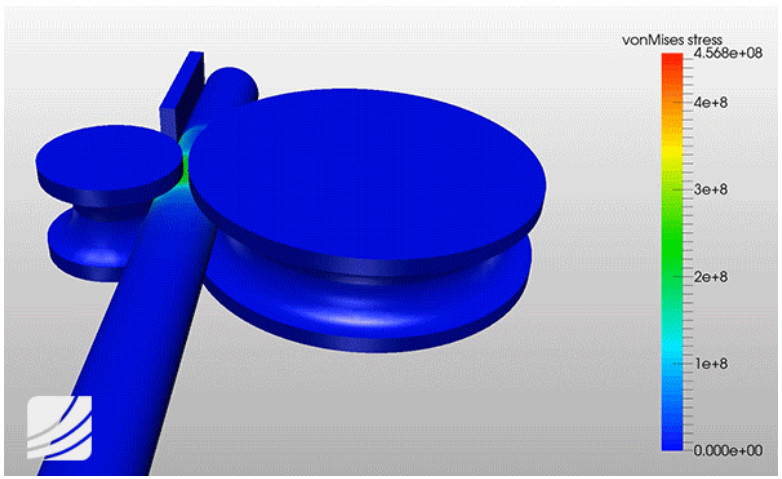
*Differential equations [...] represent the most powerful tool humanity has ever created for making sense of the material world. – Steven Strogatz (2009)*

## Setting

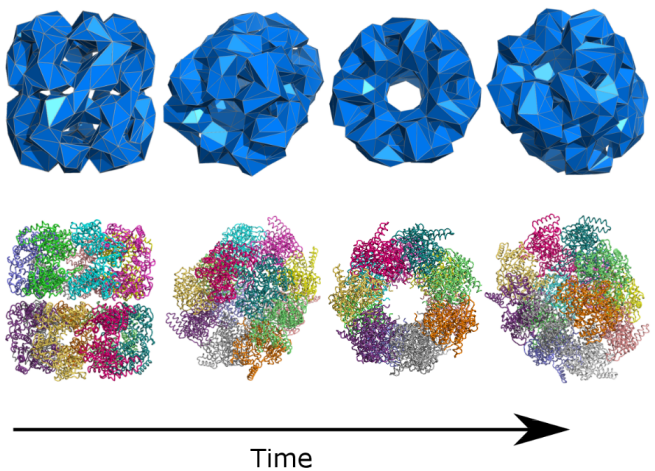
We want to numerically approximate the solution $u \in \mathcal{V}$ of a PDE

$$L_p u = f \quad \text{on } D,$$

given by a differential operator $L_p \colon \mathcal{V} \to \mathcal{V}'$ with parameters $p \in \mathcal{P}$ on a domain $D \subset \mathbb{R}^k$ and $f \in \mathcal{V}'$. Additionally, we assume that suitable boundary conditions on $\partial D$ are specified or incorporated into the function space $\mathcal{V}$.




https://commons.wikimedia.org/wiki/File:Bending_Analysis_of_an_Aluminium_Pipe.gif



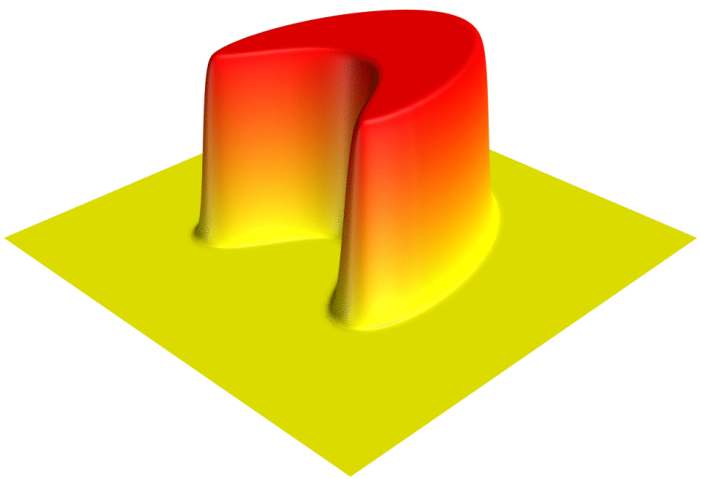

(Solernou et al., 2018)

# Examples

## Convection-diffusion equation

$$\partial_t u - \underbrace{\nabla \cdot (D \nabla u)}_{\text{diffusion}} + \underbrace{\nabla \cdot (v u)}_{\text{convection}} = \underbrace{f}_{\text{source}}$$

**Notation:** Divergence $\nabla \cdot v := \sum_i \partial_{x_i} v_i$
**Setting:** Parameters $p := (D, v)$, operator $L_p u = \partial_t u - \nabla \cdot (D \nabla u) + \nabla \cdot (v u)$
**Special case:** Heat equation for velocity $v = 0$, constant diffusivity $D$, and $f = 0$
**Applications:** Transfer of particles or energy due to diffusion and convection.



https://commons.wikimedia.org/wiki/File:Heat.gif

# Selected classical methods

## Finite difference method (FDM)

Approximate the differential operator by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x - he_i) - 2u(x) + u(x + he_i)}{h^2} \quad \Rightarrow \quad \text{linear system for } (u(x + jhe_i))_{i,j=1}^{d,n}.$$

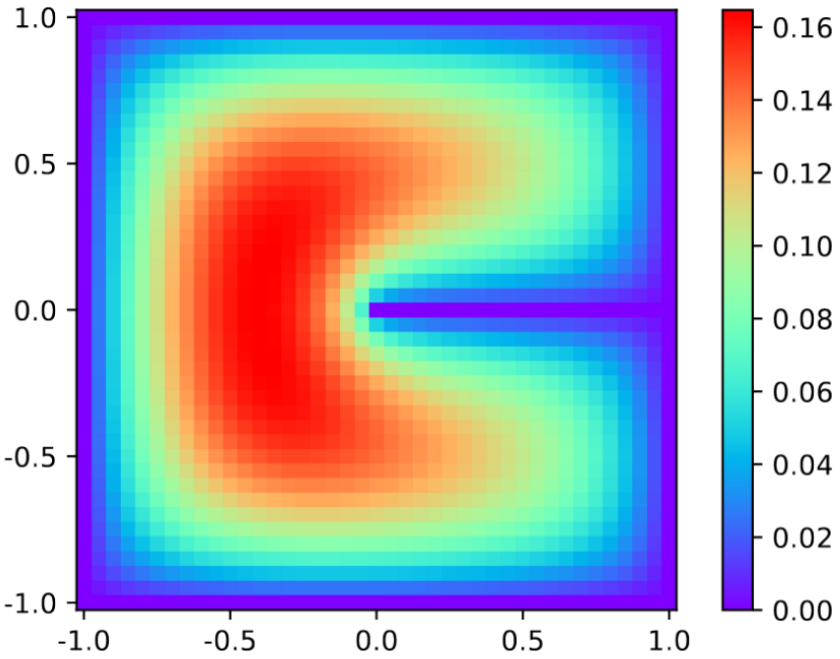



**Figure 1:** Poisson equation $\partial_{x_1 x_1} u + \partial_{x_2 x_2} u = 1$ on $D := (-1, 1)^2 \setminus [0, 1) \times \{0\}$ with $u|_{\partial D} = 0$ (E et al., 2017a).

# Selected classical methods

## Finite difference method (FDM)

Approximate the differential operator by difference quotients on a grid.

For instance:

$$\partial_{x_i x_i} u(x) \approx \frac{u(x - he_i) - 2u(x) + u(x + he_i)}{h^2} \quad \Rightarrow \quad \text{linear system for } (u(x + jhe_i))_{i,j=1}^{d,n}.$$

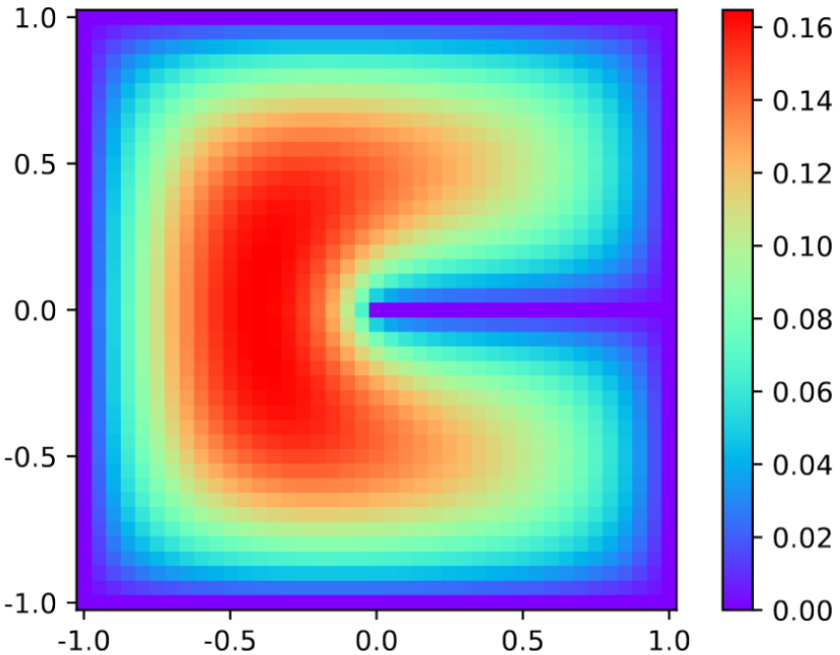



**Figure 1:** Poisson equation $\partial_{x_1 x_1} u + \partial_{x_2 x_2} u = 1$ on $D := (-1, 1)^2 \setminus [0, 1) \times \{0\}$ with $u|_{\partial D} = 0$ (E et al., 2017a).

**Method of lines:** Discretize only the spatial dimensions and use numerical solvers for ordinary differential equations (ODEs) for the time dimension.

# Selected classical methods

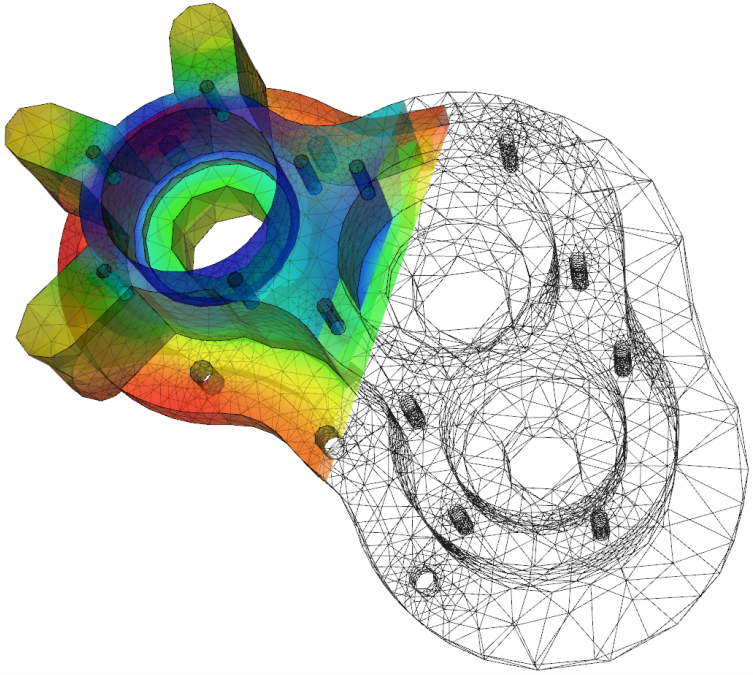
## Finite element method (FEM)

Approximate the function space $\mathcal{V}$ by a finite-dimensional space.

For instance (Galerkin approximation for linear $L_p$):

$$u(x) \approx \sum_{i=1}^{n} u_i v_i(x) \quad \Rightarrow \quad \text{linear system via } \sum_{i=1}^{n} u_i \langle L_p(v_i), v_j \rangle = \langle f, v_j \rangle, \quad j = 1, \ldots, n,$$

where $(v_i)_{i=1}^n \subset \mathcal{V}$ are piecewise polynomial basis functions with small support.



https://de.wikipedia.org/wiki/Datei:Elmer-pump-heatequation.png

# Selected classical methods

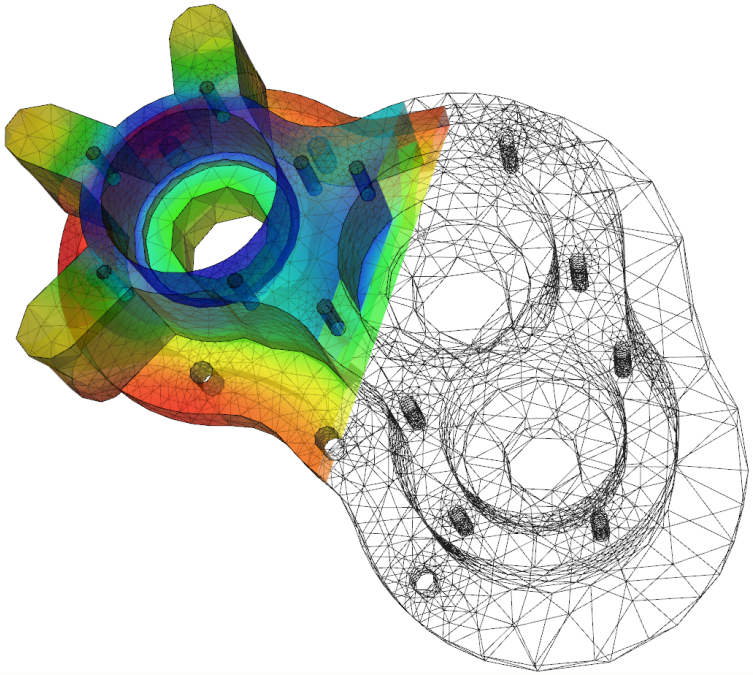
## Finite element method (FEM)

Approximate the function space $\mathcal{V}$ by a finite-dimensional space.

For instance (Galerkin approximation for linear $L_p$):

$$u(x) \approx \sum_{i=1}^{n} u_i v_i(x) \quad \Rightarrow \quad \text{linear system via } \sum_{i=1}^{n} u_i \langle L_p(v_i), v_j \rangle = \langle f, v_j \rangle, \quad j = 1, \dots, n,$$

where $(v_i)_{i=1}^n \subset \mathcal{V}$ are piecewise polynomial basis functions with small support.



https://de.wikipedia.org/wiki/Datei:Elmer-pump-heatequation.png

- 👍 high accuracies, theoretical guarantees
- 👎 needs to be tailored to the PDE, slow/infeasible for high dimensions or complicated geometries

# Overview of deep learning methods

# Surrogate models

Use training data (e.g. from classical solvers or simulations) to learn the parametric mapping from PDE parameters $p \in \mathcal{P}$ to the corresponding solution $u_p \in \mathcal{V}$.

**Idea**

1. Discretize pairs of parameters and solutions $(p^{(i)}, u^{(i)})_{i=1}^m$.
2. Use a NN $\Phi_\theta$ with parameters $\theta$ and variants of SGD to minimize

$$\min_\theta \sum_{i=1}^m \text{loss}(\Phi_\theta(p^{(i)}), u^{(i)}).$$

# Surrogate models

Use training data (e.g. from classical solvers or simulations) to learn the parametric mapping from PDE parameters $p \in \mathcal{P}$ to the corresponding solution $u_p \in \mathcal{V}$.

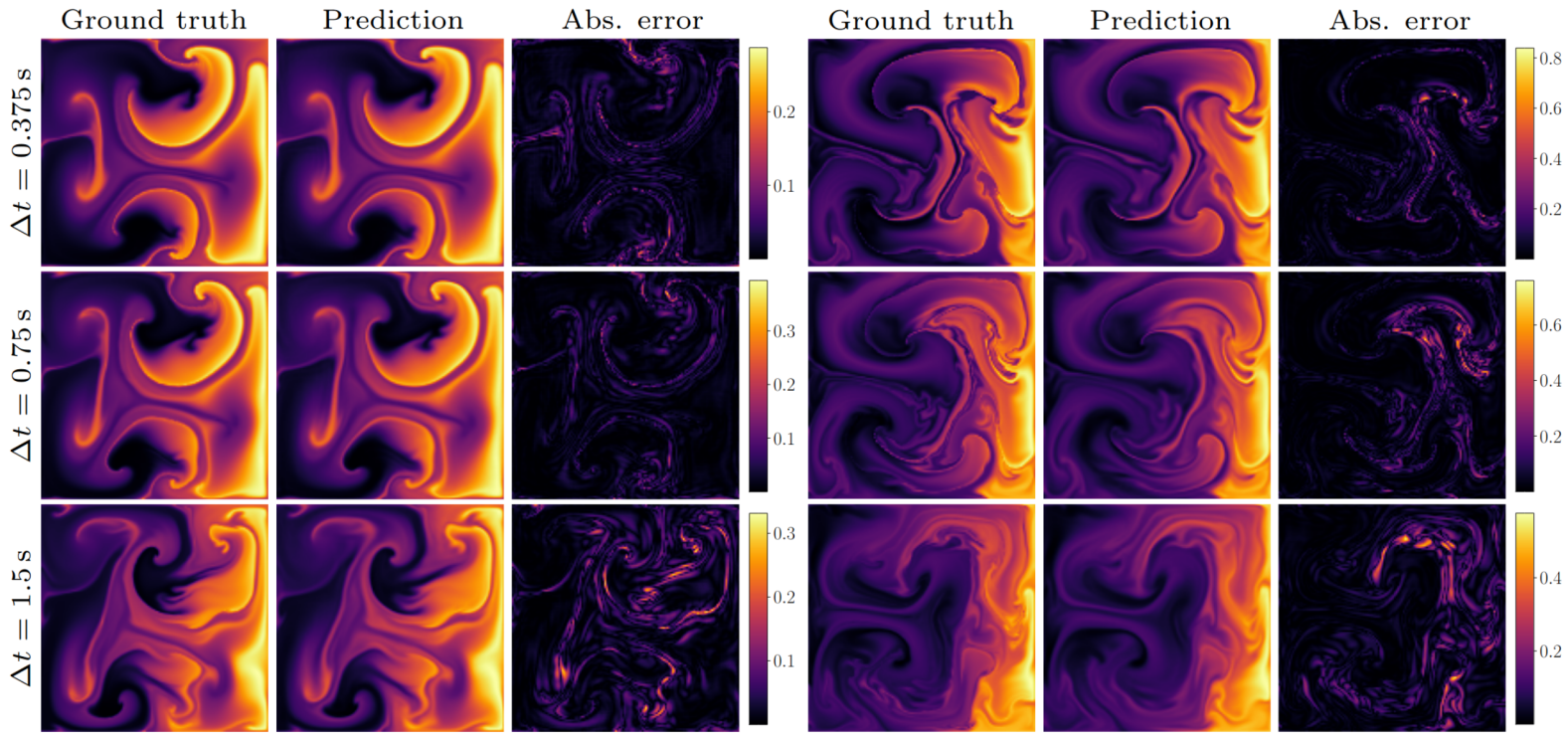
**Idea**

1. Discretize pairs of parameters and solutions $(p^{(i)}, u^{(i)})_{i=1}^m$.
2. Use a NN $\Phi_\theta$ with parameters $\theta$ and variants of SGD to minimize

$$\min_\theta \sum_{i=1}^m \text{loss}(\Phi_\theta(p^{(i)}), u^{(i)}).$$

1. Discretization of $p^{(i)}$ and $u^{(i)}$ on **fixed grids** using, e.g., U-Net, Turbulent-Flow Net, ResNet, ConvLSTM, GNN (Guo et al., 2016; Bhatnagar et al., 2019; Wang et al., 2020; Brandstetter et al., 2022; Gupta & Brandstetter, 2022).
2. **Discretization-invariant** approaches, e.g., Deep-O-Net, Neural Operators, truncated PCA expansions (Bhattacharya et al., 2020; Li et al., 2020; Kovachki et al., 2021; Lu et al., 2021; De Hoop et al., 2022).

# Surrogate models




**(a)** Buoyancy force= 0.22 **(b)** Buoyancy force= 0.48

**Figure 2:** Groundtruth fluid simulation and U-Net prediction for the particle concentration governed by a Navier-Stokes equation with two buoyancy force values (Gupta & Brandstetter, 2022).

- 👍 PDE can be unknown, fast solutions to whole families of PDEs
- 👎 needs training data

# Variational formulations / Neural-FEM

⚠ If classical methods or simulations are infeasible, we can use Neural-FEM methods to learn a solution for a fixed parameter $p \in \mathcal{P}$.

# Variational formulations / Neural-FEM

⚠ If classical methods or simulations are infeasible, we can use Neural-FEM methods to learn a solution for a fixed parameter $p \in \mathcal{P}$.

**Idea**

1. Use a variational formulation $u = \operatorname{argmin}_{v \in \mathcal{V}} \mathcal{L}(v)$ of the PDE solution.
2. Approximate the function space

$$\mathcal{V} \approx \{\Phi_\theta : \theta \in \Theta\},$$

   where $\Phi_\theta$ is a NN with parameters $\theta$.
3. Optimize (an estimator version of)

$$\min_\theta \mathcal{L}(\Phi_\theta).$$

   using variants of SGD.

# Variational formulations / Neural-FEM

⚠ If classical methods or simulations are infeasible, we can use Neural-FEM methods to learn a solution for a fixed parameter $p \in \mathcal{P}$.

**Idea**

1. Use a variational formulation $u = \operatorname{argmin}_{v \in \mathcal{V}} \mathcal{L}(v)$ of the PDE solution.
2. Approximate the function space

$$\mathcal{V} \approx \{\Phi_\theta : \theta \in \Theta\},$$

   where $\Phi_\theta$ is a NN with parameters $\theta$.
3. Optimize (an estimator version of)

$$\min_\theta \mathcal{L}(\Phi_\theta).$$

   using variants of SGD.

In Item 2 one can try to incorporate prior knowledge (smoothness, boundary values, symmetries, conservation laws) into the architecture of $\Phi_\theta$.

# Examples for variational formulations

**Reminder:** We want to solve $L_p u = f$ on the domain $D$.

**Deep-Galerkin method (Sirignano & Spiliopoulos, 2018) & Physics-informed NNs (Raissi et al., 2019)**

$$\mathcal{L}_D(v) := \mathbb{E}_{x \sim \mathrm{Unif}(D)}\left[|(L_p v)(x) - f(x)|^2\right].$$

# Examples for variational formulations

**Reminder:** We want to solve $L_p u = f$ on the domain $D$.

**Deep-Galerkin method (Sirignano & Spiliopoulos, 2018) & Physics-informed NNs (Raissi et al., 2019)**

$$\mathcal{L}_D(v) := \mathbb{E}_{x \sim \mathrm{Unif}(D)}\left[|(L_p v)(x) - f(x)|^2\right].$$

**Implementation:** Compute $L_p \Phi_\theta$ using automatic differentiation and minimize

$$\min_\theta \sum_{i=1}^{n} |(L_p \Phi_\theta)(x^{(i)}) - f(x^{(i)})|^2, \quad x^{(i)} \sim \mathrm{Unif}(D)$$

using gradient descent.

# Examples for variational formulations

**Extensions:**

- Use PDE-specific distributions to sample the domain.

# Examples for variational formulations

**Extensions:**

- Use PDE-specific distributions to sample the domain.
- Add boundary conditions $u|_{\partial D} = g$ and known data points $(\bar{x}^{(i)}, u(\bar{x}^{(i)}))_{i=1}^{m}$ as penalty terms

$$\mathcal{L} := \mathcal{L}_D + \lambda_1 \mathcal{L}_{\partial D} + \lambda_2 \mathcal{L}_{\text{data}}$$

where

$$\mathcal{L}_{\partial D}(v) := \mathbb{E}_{x \sim \text{Unif}(\partial D)}\left[|v(x) - g(x)|^2\right], \quad \mathcal{L}_{\text{data}}(v) := \sum_{i=1}^{m} |v(\bar{x}^{(i)}) - u(\bar{x}^{(i)})|^2.$$

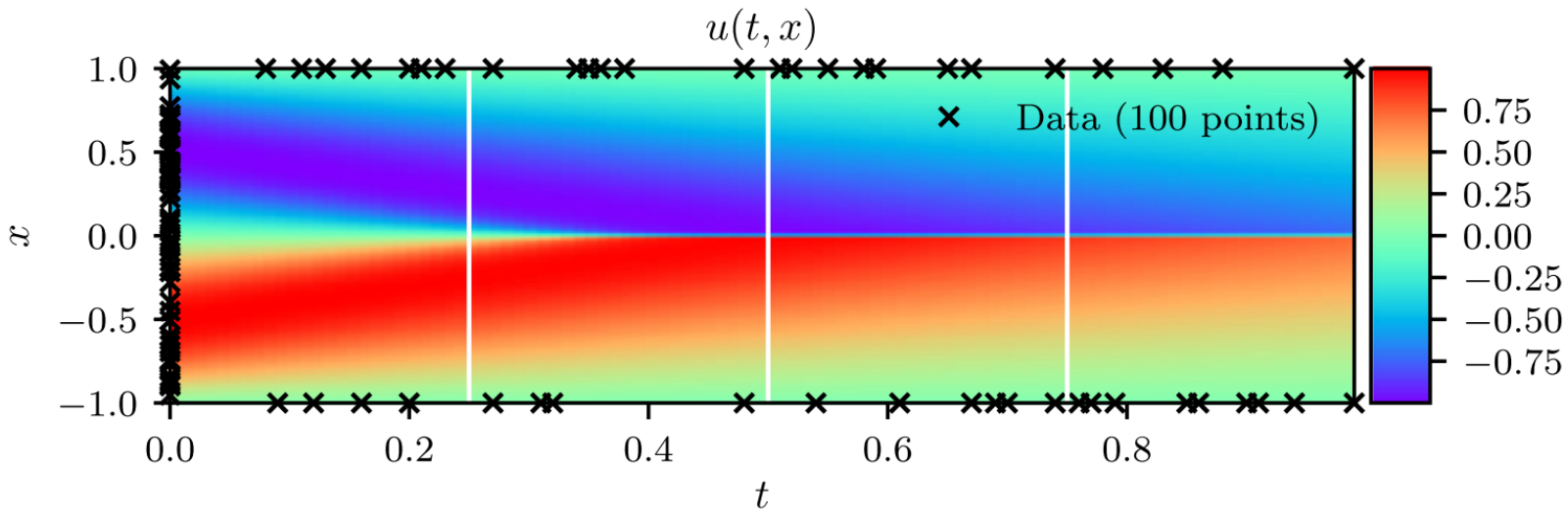



**Figure 3:** Burger's equation $\partial_t u + u\partial_x u - \frac{0.01}{\pi}\partial_{xx} u = 0$ (Raissi et al., 2019).

# Examples for variational formulations

**Variational formulation**

$$\mathcal{L}(v) \coloneqq \mathbb{E}_{x\sim\mathrm{Unif}(D)}\left[|(L_p v)(x) - f(x)|^2\right] + \lambda_1 \mathcal{L}_{\partial D}(v) + \lambda_2 \mathcal{L}_{\mathrm{data}}(v)$$

- 👍 applicable to most PDEs, unlimited data, no classical solvers needed, mesh-free
- 👎 sensitive to hyperparameters $\lambda_1, \lambda_2$ and sampling schemes, higher-order derivatives $\partial_\theta L_p(\Phi_\theta)$ (can be reduced with weak formulations of PDEs, e.g., Deep-Ritz (E et al., 2017b) and Friedrichs Learning (Chen et al., 2020))

⚠ For specific types of PDEs we can introduce variational formulations **without higher-order derivatives** and include boundary conditions **without penalty terms** (E et al., 2017a; Berner, Dablander, et al., 2020; Beck et al., 2021).

# SDE-based variational formulations

# Stochastic differential equations (SDEs)

**Applications:** Modelling random processes, e.g., in physics and chemistry (Langevin equation) or financial engineering (Black-Scholes model).

**Setting**

We consider solution processes $X$ to SDEs of the form

$$\mathrm{d}X_s = \underbrace{\mu(X_s, s)}_{\text{drift}}\,\mathrm{d}s + \underbrace{\sigma(X_s, s)}_{\text{diffusion}}\,\mathrm{d}B_s,$$

where $B$ is a $d$-dim. Brownian motion.

# Stochastic differential equations (SDEs)

**Applications:** Modelling random processes, e.g., in physics and chemistry (Langevin equation) or financial engineering (Black-Scholes model).

**Setting**

We consider solution processes $X$ to SDEs of the form

$$\mathrm{d}X_s = \underbrace{\mu(X_s, s)}_{\text{drift}}\,\mathrm{d}s + \underbrace{\sigma(X_s, s)}_{\text{diffusion}}\,\mathrm{d}B_s,$$

where $B$ is a $d$-dim. Brownian motion.

Intuition via the Euler-Maruyama scheme $\hat{X}_t \approx X_t$:

$$\hat{X}_{t+\Delta t} = \hat{X}_t + \mu(\hat{X}_t, t)\Delta t + \sigma(\hat{X}_t, t)\underbrace{(B_{t+\Delta t} - B_t)}_{\sim\mathcal{N}(0,\Delta t)}.$$


**Figure 4:** SDE solution and Euler-Maruyama scheme with $\Delta t = \frac{1}{N}$ and $N = 4, 8$.

# SDE-based representation

**Kolmogorov backwards equation and Feynman-Kac formula**

$$\begin{cases} \partial_t u + \frac{1}{2}\operatorname{tr}(\sigma\sigma^\top \operatorname{Hess}_u) + \partial_x u \mu = 0 & \text{on } \mathbb{R}^d \\ u(\cdot, T) = \varphi & \text{on } \mathbb{R}^d. \end{cases}$$

Then

$$u(x,t) = \mathbb{E}\left[\varphi(X_T) | X_t = x\right].$$

**Applications:** Black-Scholes and Heston model (financial engineering), Hamilton-Jacobi-Bellman equation (via Hopf-Cole transformation), stochastic Lorenz system, and heat equation ($\sigma = \mathrm{I}$ and $\mu = 0$).

# SDE-based representation

**Kolmogorov backwards equation and Feynman-Kac formula**

$$\begin{cases} \partial_t u + \frac{1}{2}\operatorname{tr}(\sigma\sigma^\top \operatorname{Hess}_u) + \partial_x u \mu = 0 & \text{on } \mathbb{R}^d \\ u(\cdot, T) = \varphi & \text{on } \mathbb{R}^d. \end{cases}$$

Then

$$u(x,t) = \mathbb{E}\left[\varphi(X_T)|X_t = x\right].$$

**Applications:** Black-Scholes and Heston model (financial engineering), Hamilton-Jacobi-Bellman equation (via Hopf-Cole transformation), stochastic Lorenz system, and heat equation ($\sigma = \mathrm{I}$ and $\mu = 0$).

**Proof:** Itô's lemma (generalization of the chain rule) shows that

$$\underbrace{u(X_T, T)}_{=\varphi(X_T)} = \underbrace{u(X_t, t)}_{=u(x,t)} + \int_t^T \big(\underbrace{\partial_t u + \tfrac{1}{2}\operatorname{tr}(\sigma\sigma^\top \operatorname{Hess}_u) + \partial_x u \mu}_{=0}\big)(X_s, s)\mathrm{d}s + S,$$

where

$$S = \int_t^T \big(\partial_x u \sigma\big)(X_s, s)\mathrm{d}B_s \quad \text{with} \quad \mathbb{E}[S|X_t = x] = 0.$$

# Generalized representation theorem

There exist versions for bounded domains, elliptic, and non-linear problems (e.g., Baldi, 2017; E et al., 2017a).

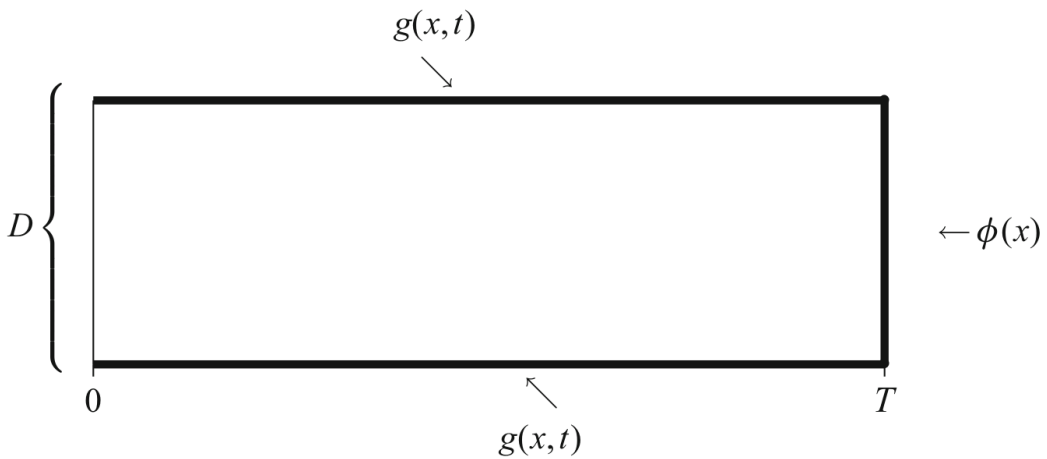
**Generalized Kolmogorov backwards equation and Feynman-Kac formula**

$$\begin{cases} \partial_t u + \frac{1}{2}\operatorname{tr}(\sigma\sigma^\top \partial_{xx} u) + \partial_x u \mu - cu = f & \text{on } D \\ u(\cdot, T) = \varphi & \text{on } D \\ u = g & \text{on } \partial D \times [0, T]. \end{cases}$$

Then

$$u(x,t) = \mathbb{E}\left[ g(X_\tau, \tau) E_\tau \mathbf{1}_{\{\tau < T\}} + \varphi(X_T) E_T \mathbf{1}_{\{\tau \geq T\}} - \int_t^{\min\{\tau, T\}} f(X_s, s) E_s \mathrm{d}s \,\middle|\, X_t = x \right]$$

where $E_s = e^{-\int_t^s c(X_u, u)\mathrm{d}u}$ and $\tau = \min\{t \in [0, \infty) : X_t \in \partial D\}$.

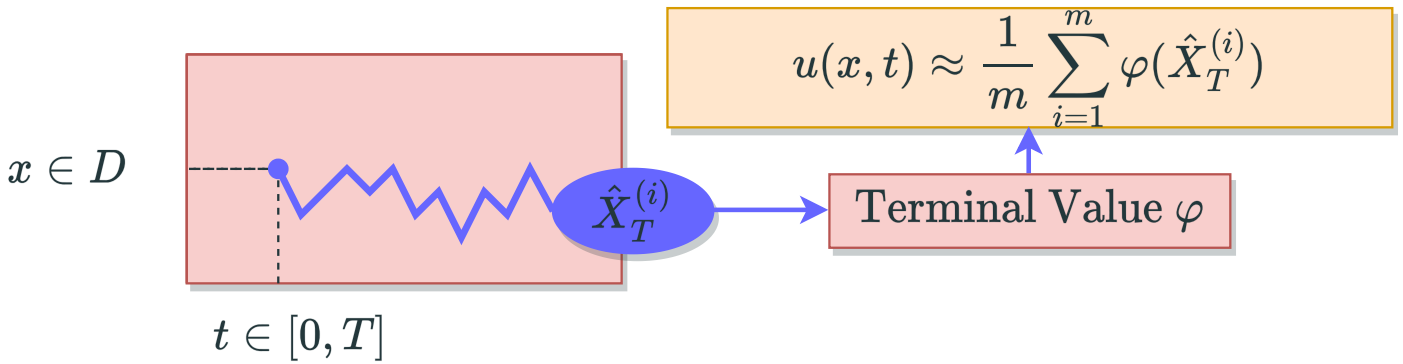


# Monte-Carlo method

**Reminder (standard Feynman-Kac formula):** $u(x,t) = \mathbb{E}\left[\varphi(X_T)|X_t = x\right]$.

# Monte-Carlo method

**Reminder (standard Feynman-Kac formula):** $u(x,t) = \mathbb{E}\left[\varphi(X_T)|X_t = x\right]$.

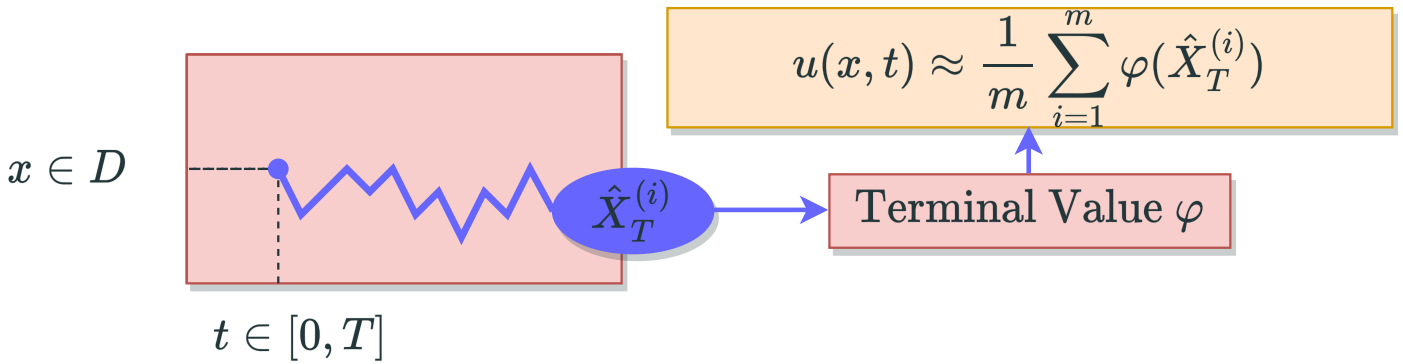



Monte-Carlo method for **pointwise** evaluation of the solution $u$:

1. Simulate i.i.d. $\hat{X}_T^{(i)}$ with $\hat{X}_t^{(i)} = x$ using the Euler-Maruyama scheme.
2. Compute $u(x,t) \approx \frac{1}{m}\sum_{i=1}^{m}\varphi(\hat{X}_T^{(i)})$.

# Monte-Carlo method

**Reminder (standard Feynman-Kac formula):** $u(x,t) = \mathbb{E}\left[\varphi(X_T)|X_t = x\right]$.

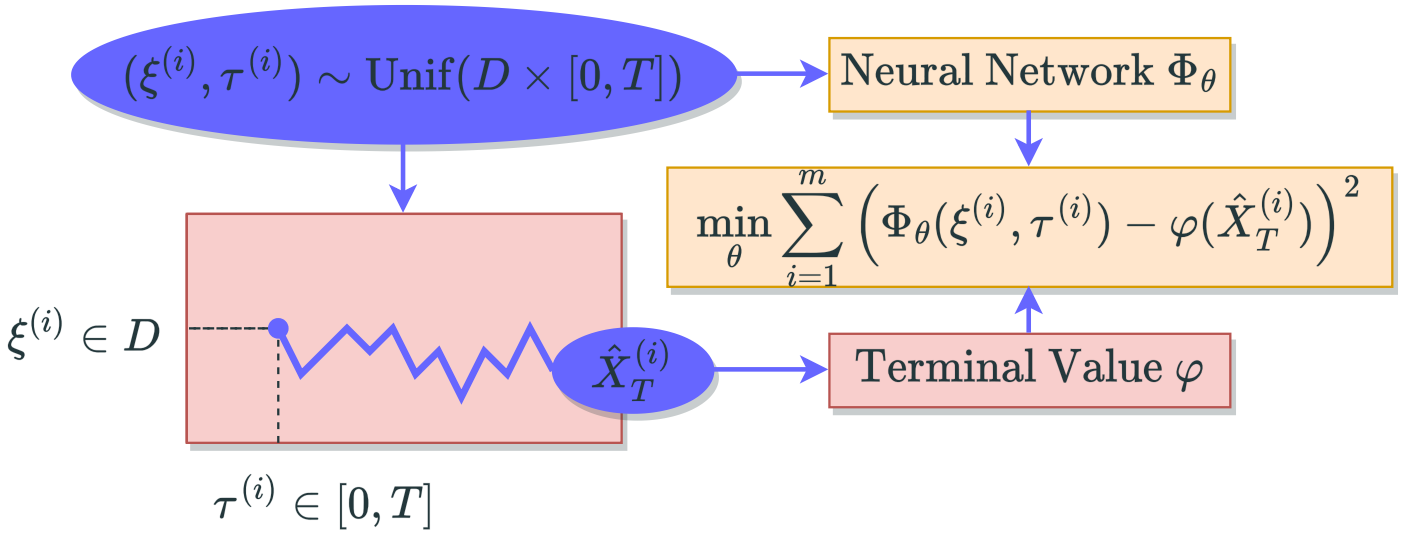



Monte-Carlo method for **pointwise** evaluation of the solution $u$:

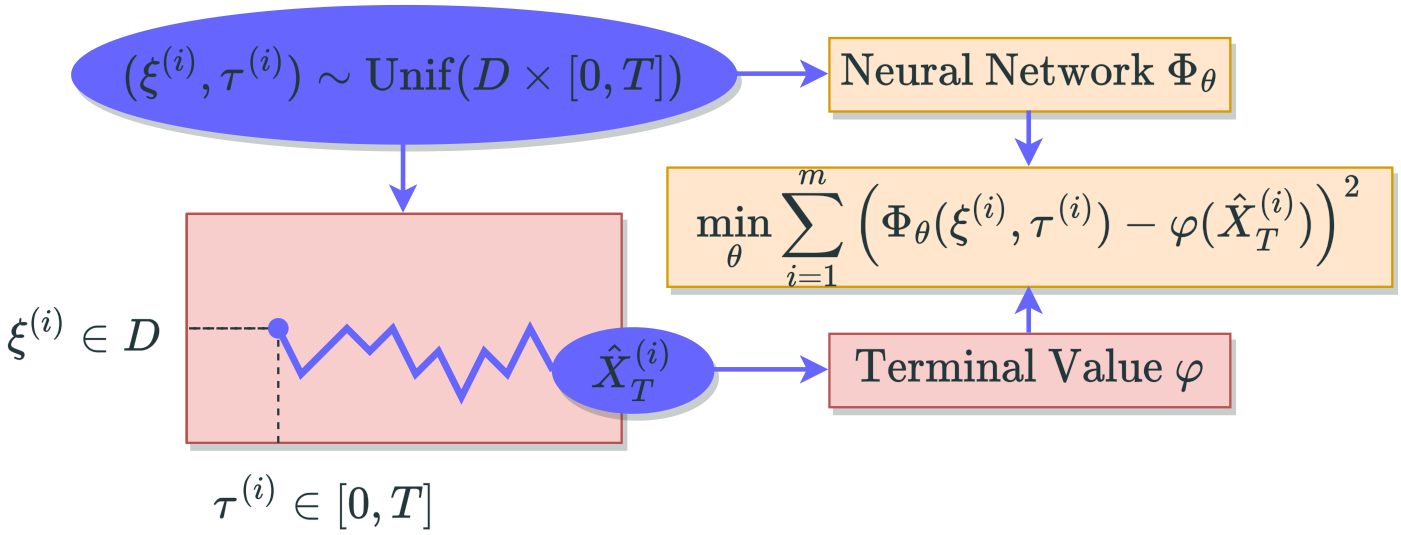
1. Simulate i.i.d. $\hat{X}_T^{(i)}$ with $\hat{X}_t^{(i)} = x$ using the Euler-Maruyama scheme.
2. Compute $u(x,t) \approx \frac{1}{m}\sum_{i=1}^{m}\varphi(\hat{X}_T^{(i)})$.

⚠ We want to solve the PDE on a **domain**, i.e., for all $(x,t) \in D \times [0,T]$.

# SDE-based variational formulation



# SDE-based variational formulation

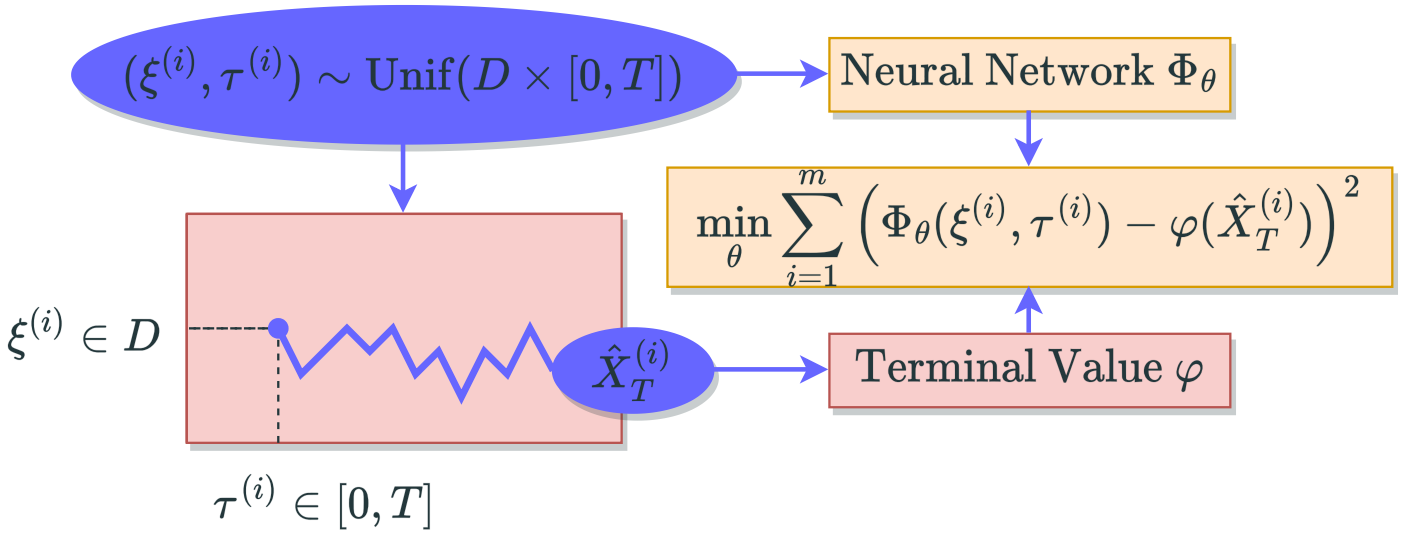



**Variational formulation (Berner, Dablander, et al., 2020; Beck et al., 2021)**

Define the target $Y = \varphi(X_T)$ where $X_\tau = \xi$ with $(\xi, \tau) \sim \mathrm{Unif}(D \times [0, T])$. Then $u$ is the unique minimizer of

$$\mathcal{L}_{FK}(v) = \mathbb{E}\left[(v(\xi, \tau) - Y)^2\right].$$

# SDE-based variational formulation




**Variational formulation (Berner, Dablander, et al., 2020; Beck et al., 2021)**

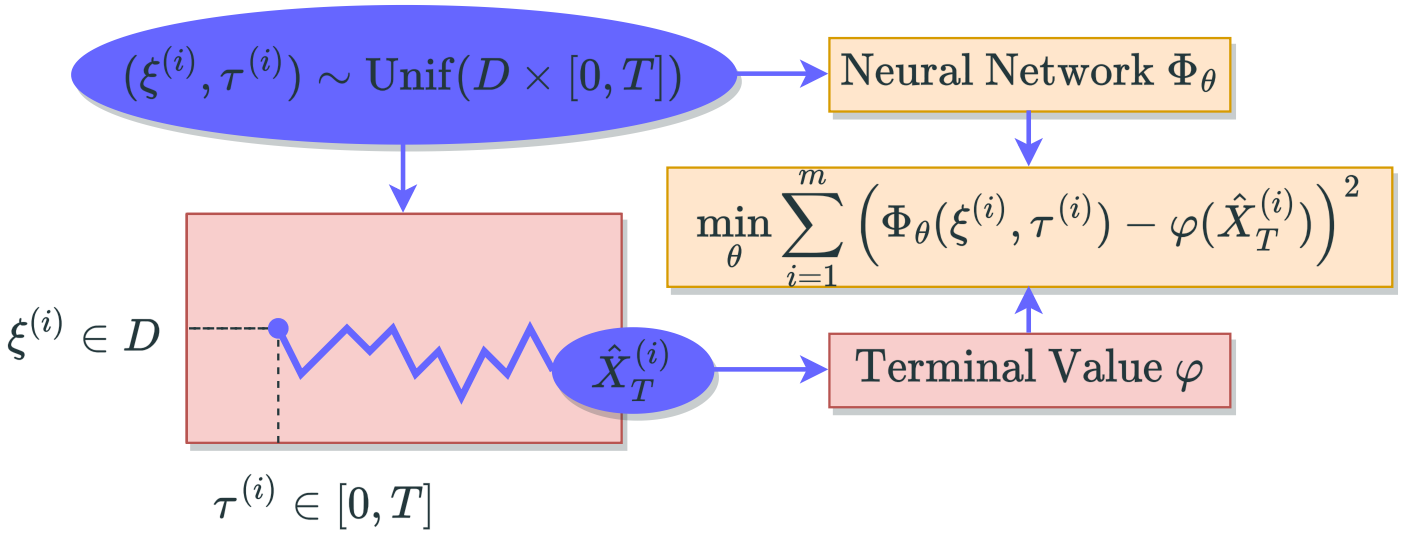
Define the target $Y = \varphi(X_T)$ where $X_\tau = \xi$ with $(\xi, \tau) \sim \mathrm{Unif}(D \times [0, T])$. Then $u$ is the unique minimizer of

$$\mathcal{L}_{FK}(v) = \mathbb{E}\left[(v(\xi, \tau) - Y)^2\right].$$

**Proof:** For almost all $(x, t) \in D \times [0, T]$ the regression function can be computed as

$$\mathbb{E}\left[Y | (\xi, \tau) = (x, t)\right] = \mathbb{E}\left[\varphi(X_T) | X_t = x\right] = u(x, t).$$

# SDE-based variational formulation




**Variational formulation (Berner, Dablander, et al., 2020; Beck et al., 2021)**

Define the target $Y = \varphi(X_T)$ where $X_\tau = \xi$ with $(\xi, \tau) \sim \mathrm{Unif}(D \times [0, T])$. Then $u$ is the unique minimizer of

$$\mathcal{L}_{FK}(v) = \mathbb{E}\left[(v(\xi, \tau) - Y)^2\right].$$

**Proof:** For almost all $(x, t) \in D \times [0, T]$ the regression function can be computed as

$$\mathbb{E}\left[Y | (\xi, \tau) = (x, t)\right] = \mathbb{E}\left[\varphi(X_T) | X_t = x\right] = u(x, t).$$

**Extensions** to parametric families of PDEs with $\mu_\gamma$, $\sigma_\gamma$, and $\varphi_\gamma$ depending on a parameter $\gamma$ (Berner, Dablander, et al., 2020).

# Performance and guarantees

# Overview

## SDE-based variational formulation

$$\mathcal{L}_{FK}(v) = \mathbb{E}\left[(v(\xi,\tau) - Y)^2\right], \quad Y = \varphi(X_T), \quad X_\tau = \xi, \quad (\xi,\tau) \sim \mathrm{Unif}(D \times [0,T]).$$

⚠ Works efficiently in dimension which are out of scope for classical methods:

| step | avg. time (s) | avg. rel. $\mathcal{L}^1$-error |
|---|---|---|
| 0 | 0 | $0.7912 \pm 0.0276$ |
| 12k | 2434 | $0.0062 \pm 0.0009$ |
| 28k | 6024 | $0.0039 \pm 0.0001$ |

**Table 1:** Solving a 53-dim. parametric option pricing problem (Berner, Dablander, et al., 2020).

# Overview

## SDE-based variational formulation

$$\mathcal{L}_{FK}(v) = \mathbb{E}\left[(v(\xi,\tau) - Y)^2\right], \quad Y = \varphi(X_T), \quad X_\tau = \xi, \quad (\xi,\tau) \sim \mathrm{Unif}(D \times [0,T]).$$

⚠ Works efficiently in dimension which are out of scope for classical methods:

| step | avg. time (s) | avg. rel. $\mathcal{L}^1$-error |
|---|---|---|
| 0 | 0 | $0.7912 \pm 0.0276$ |
| 12k | 2434 | $0.0062 \pm 0.0009$ |
| 28k | 6024 | $0.0039 \pm 0.0001$ |

**Table 1:** Solving a 53-dim. parametric option pricing problem (Berner, Dablander, et al., 2020).

We can provide justifications by analyzing the size of the architecture of $\Phi_\theta$ and the number of samples $m$ needed such that

$$\mathbb{E}\left[\left(\Phi_{\theta^*}(\xi,\tau) - u(\xi,\tau)\right)^2\right] \leq \varepsilon,$$

where $\theta^*$ is an empirical risk minimizer:

$$\theta^* \in \operatorname{argmin} \sum_{i=1}^{m} \left(\Phi_\theta(\xi^{(i)},\tau^{(i)}) - Y^{(i)}\right)^2.$$

# Approximation and generalization error

**Theorem (informal) (Jentzen et al., 2018; Berner, Grohs, & Jentzen, 2020; Grohs & Herrmann, 2020; Reisinger & Zhang, 2020)**

Assume that $\varphi$, $\mu$, and $\sigma$ can be efficiently approximated by NNs. Then, with high probability, any empirical risk minimizer $\theta^*$ satisfies that

$$\mathbb{E}\left[\left(\Phi_{\theta^*}(\xi,\tau) - u(\xi,\tau)\right)^2\right] \leq \varepsilon,$$

where the **size of the architecture** and the **number of samples** only **scale polynomially in the dimension** $d$ and $\varepsilon^{-1}$.

# Approximation and generalization error

**Theorem (informal) (Jentzen et al., 2018; Berner, Grohs, & Jentzen, 2020; Grohs & Herrmann, 2020; Reisinger & Zhang, 2020)**

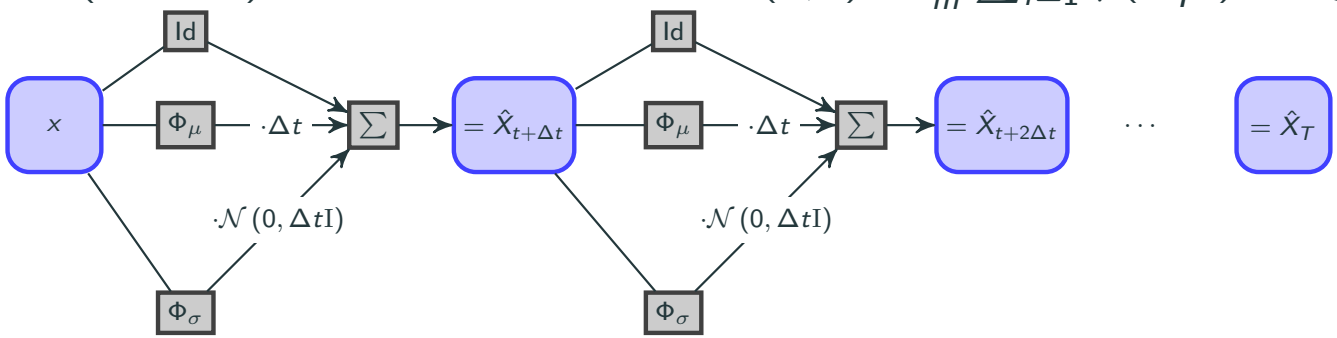
Assume that $\varphi$, $\mu$, and $\sigma$ can be efficiently approximated by NNs. Then, with high probability, any empirical risk minimizer $\theta^*$ satisfies that

$$\mathbb{E}\left[\left(\Phi_{\theta^*}(\xi,\tau) - u(\xi,\tau)\right)^2\right] \leq \varepsilon,$$

where the **size of the architecture** and the **number of samples** only **scale polynomially in the dimension** $d$ and $\varepsilon^{-1}$.

**Proof idea (approximation):**

1. Reminder: $u(x,t) = \mathbb{E}\left[\varphi(X_T)|X_t = x\right]$.
2. Emulate Euler-Maruyama scheme using NNs $\Phi_\mu \approx \mu$, $\Phi_\sigma \approx \sigma$.
3. Compose with a NN $\Phi_\varphi \approx \varphi$ to obtain an approximation of $\varphi(\hat{X}_T)$.
4. Emulate (uniform) Monte-Carlo estimator $u(x,t) \approx \frac{1}{m}\sum_{i=1}^m \varphi(\hat{X}_T^{(i)})$ using NNs.




**Proof idea (generalization):** Covering numbers and concentration inequalities.

# Optimization error

⚠ Does SGD approximately yield an empirical risk minimizer $\theta^*$?

- In restricted settings there exist guarantees, for instance, via the lazy regime or a loss landscape analysis, see, e.g., Berner et al., 2019; Berner, Grohs, et al., 2022.

# Optimization error

⚠ Does SGD approximately yield an empirical risk minimizer $\theta^*$?

- In restricted settings there exist guarantees, for instance, via the lazy regime or a loss landscape analysis, see, e.g., Berner et al., 2019; Berner, Grohs, et al., 2022.
- Not directly applicable to our problem, but promising empirical results:

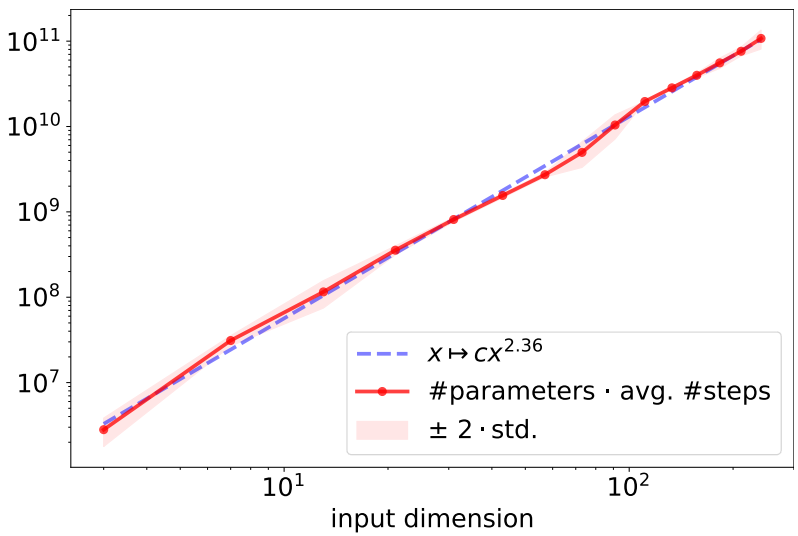



**Figure 5:** Cost for solving a $d$-dim. heat equation up to a relative $\mathcal{L}^1$-error of $10^{-2}$ (Berner, Dablander, et al., 2020).

# Sticking-the-landing estimator

**Theorem (informal) (Richter & Berner, 2022)**

Subtracting the stochastic integral $\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s$ from the loss yields (gradient) **estimators with vanishing variance** when converging to the optimum.

# Sticking-the-landing estimator

**Theorem (informal) (Richter & Berner, 2022)**

Subtracting the stochastic integral $\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s$ from the loss yields (gradient) **estimators with vanishing variance** when converging to the optimum.

**Proof Idea:** By Itô's lemma it holds that

$$\mathbb{V}\left[(u(\xi,\tau) - \varphi(X_T))^2\right] = \mathbb{V}\left[\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s\right].$$

If $\Phi_\theta \approx u$, the variance of the (gradient) estimator of our loss is non-zero.

# Sticking-the-landing estimator

**Theorem (informal) (Richter & Berner, 2022)**

Subtracting the stochastic integral $\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s$ from the loss yields (gradient) **estimators with vanishing variance** when converging to the optimum.

**Proof Idea:** By Itô's lemma it holds that

$$\mathbb{V}\left[(u(\xi,\tau) - \varphi(X_T))^2\right] = \mathbb{V}\left[\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s\right].$$

If $\Phi_\theta \approx u$, the variance of the (gradient) estimator of our loss is non-zero.
**Solution:** Define the loss

$$\mathcal{L}_{\mathrm{BSDE}}(v) = \mathbb{E}\left[\left(v(\xi,\tau) - \varphi(X_T) - \int_\tau^T (\partial_x v\sigma)(X_s, s)\,\mathrm{d}B_s\right)^2\right].$$

# Sticking-the-landing estimator

**Theorem (informal) (Richter & Berner, 2022)**

Subtracting the stochastic integral $\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s$ from the loss yields (gradient) **estimators with vanishing variance** when converging to the optimum.

**Proof Idea:** By Itô's lemma it holds that

$$\mathbb{V}\left[(u(\xi,\tau) - \varphi(X_T))^2\right] = \mathbb{V}\left[\int_\tau^T (\partial_x u\sigma)(X_s, s)\,\mathrm{d}B_s\right].$$

If $\Phi_\theta \approx u$, the variance of the (gradient) estimator of our loss is non-zero.
**Solution:** Define the loss

$$\mathcal{L}_{\mathrm{BSDE}}(v) = \mathbb{E}\left[\left(v(\xi,\tau) - \varphi(X_T) - \int_\tau^T (\partial_x v\sigma)(X_s, s)\,\mathrm{d}B_s\right)^2\right].$$

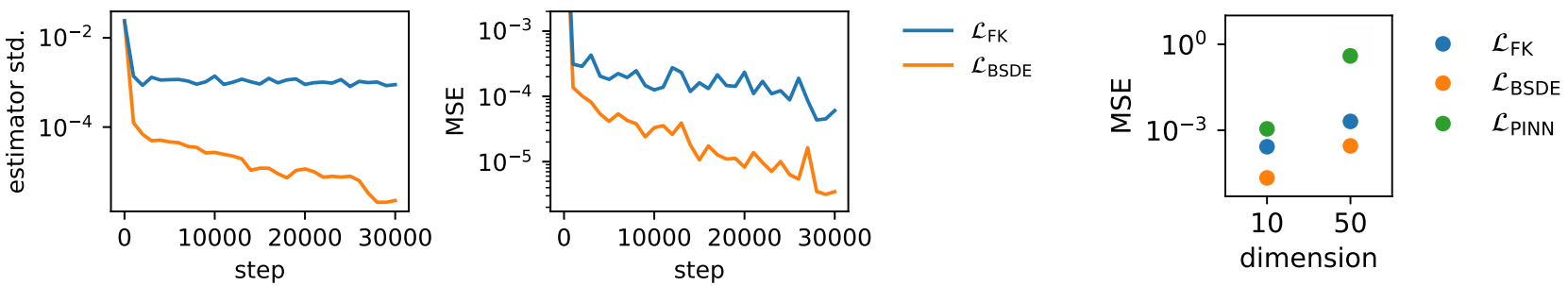



**Figure 6:** Comparisons for 10-dim. HJB equation and 10-/50-dim. heat equation.

# Connection to diffusion models

# Sampling from high-dimensional distributions

**Task**

Sample from a high-dimensional distribution $Y$.

# Sampling from high-dimensional distributions

**Task**

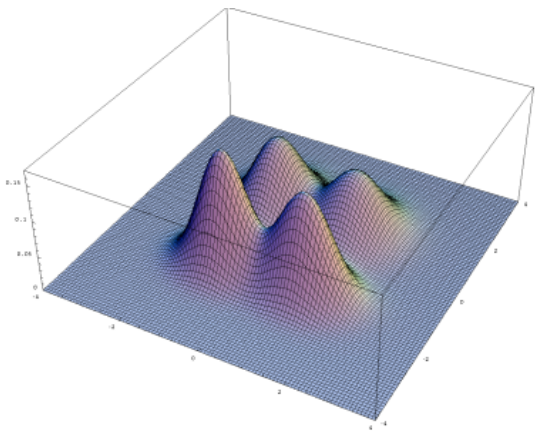
Sample from a high-dimensional distribution $Y$.

$Y$ can be given in the form of:

1. **samples** $Y^{(i)} \sim Y$ (images, text, sound, ...).

https://en.m.wikipedia.org/wiki/File:Cat_poster_1.jpg

2. an (unnormalized) **density** $\rho$ with $p_Y = \rho/\mathcal{Z}$ (e.g., in Bayesian statistics, computational physics, and chemistry).



https://en.wikipedia.org/wiki/File:Bimodal-bivariate-small.png

# Diffusion models

Established themselves as **start-of-the art in generative modeling and likelihood estimation** of image data (Song et al., 2020; Kingma et al., 2021; Nichol & Dhariwal, 2021).

# Diffusion models

Established themselves as **start-of-the art in generative modeling and likelihood estimation** of image data (Song et al., 2020; Kingma et al., 2021; Nichol & Dhariwal, 2021).

**Figure 7:** Sampling conditioned on the text prompt "a photograph of an astronaut riding a horse" using the stable diffusion model (Rombach et al., 2021).

# Applications in chemistry and biology

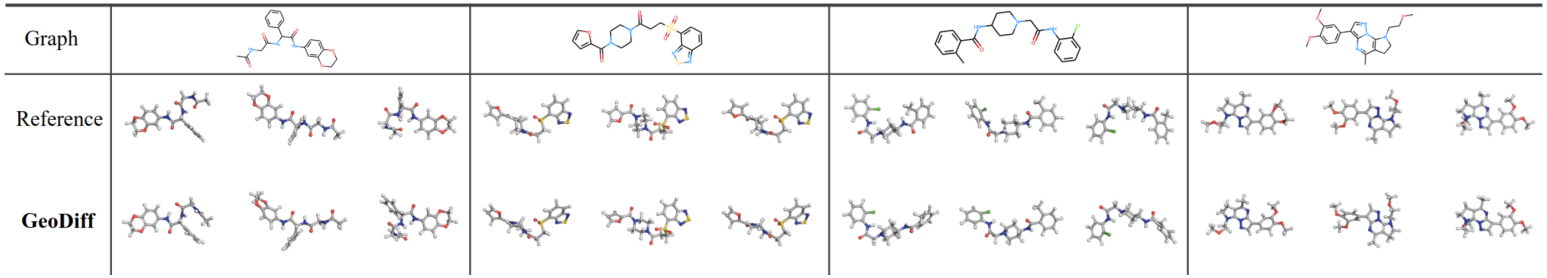



**Figure 8:** Molecular conformer generation (Xu et al., 2022). Follow-up work by (Jing et al., 2022) outperforms state-of-the-art cheminformatics methods.

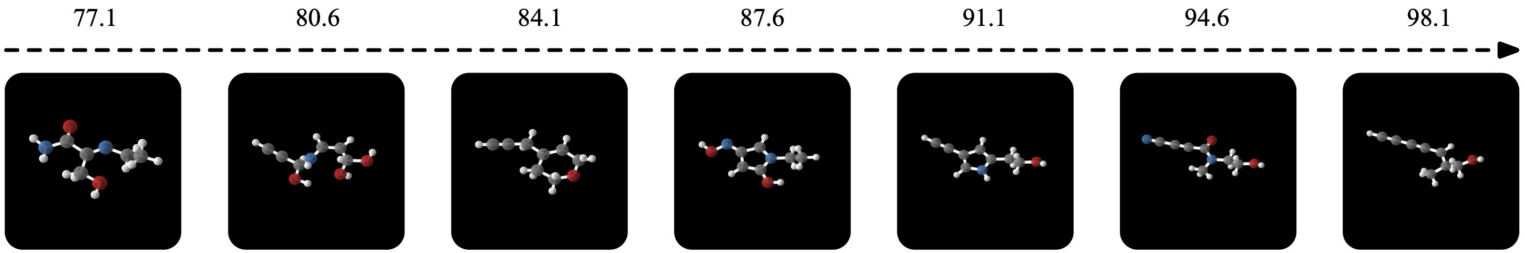



**Figure 9:** Molecule generation conditioned on polarizability (Hoogeboom et al., 2022).

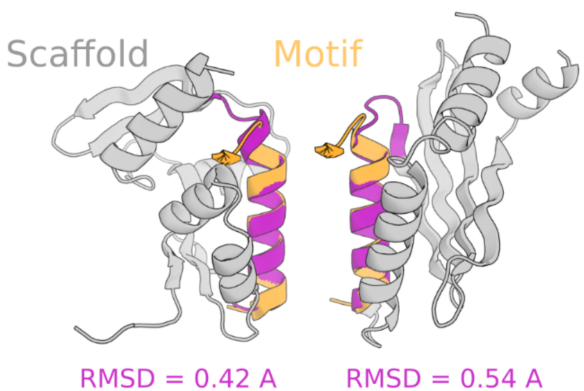



**Figure 10:** Generation of protein backbones (Trippe et al., 2022).

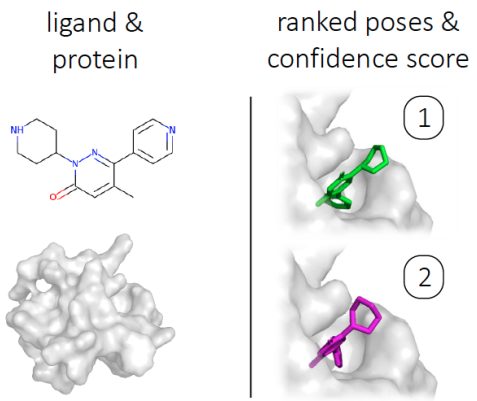



**Figure 11:** State-of-the-art molecular docking methods (Corso et al., 2022; Qiao et al., 2022).

# Theoretical foundations

Let the SDE (typically an Ornstein–Uhlenbeck process)

$$\mathrm{d}Y_s = \mu(Y_s, s)\mathrm{d}s + \sigma(s)\mathrm{d}B_s$$

diffuse the data $Y_0$ to $Y_T \approx \mathcal{N}(0, \mathrm{I})$.

(Nichol & Dhariwal, 2021)

# Theoretical foundations

Let the SDE (typically an Ornstein–Uhlenbeck process)

$$\mathrm{d}Y_s = \mu(Y_s, s)\mathrm{d}s + \sigma(s)\mathrm{d}B_s$$

diffuse the data $Y_0$ to $Y_T \approx \mathcal{N}(0, \mathrm{I})$.

(Nichol & Dhariwal, 2021)

We can reverse the diffusion:

**Reverse-time generative SDE (Anderson, 1982; Song et al., 2020)**

The solution to the SDE

$$\mathrm{d}X_s = \left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla \log \overleftarrow{p}_{Y_s} - \overleftarrow{\mu}\right)(X_s, s)\mathrm{d}s + \overleftarrow{\sigma}(s)\mathrm{d}B_s, \quad X_0 \sim Y_T,$$

satisfies that $X_s \sim Y_{T-s}$, where $p_{Y_s}$ denotes the density of $Y_s$ and where we use the notation $\overleftarrow{f}(t) = f(T-t)$.

# Theoretical foundations

Let the SDE (typically an Ornstein–Uhlenbeck process)

$$\mathrm{d}Y_s = \mu(Y_s, s)\mathrm{d}s + \sigma(s)\mathrm{d}B_s$$

diffuse the data $Y_0$ to $Y_T \approx \mathcal{N}(0, \mathrm{I})$.

(Nichol & Dhariwal, 2021)

We can reverse the diffusion:

**Reverse-time generative SDE (Anderson, 1982; Song et al., 2020)**

The solution to the SDE

$$\mathrm{d}X_s = \left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla \log \overleftarrow{p}_{Y_s} - \overleftarrow{\mu}\right)(X_s, s)\mathrm{d}s + \overleftarrow{\sigma}(s)\mathrm{d}B_s, \quad X_0 \sim Y_T,$$

satisfies that $X_s \sim Y_{T-s}$, where $p_{Y_s}$ denotes the density of $Y_s$ and where we use the notation $\overleftarrow{f}(t) = f(T-t)$.

**Proof:** Show that $\overleftarrow{p}_{Y_s}$ and $p_{X_s}$ satisfy the same Fokker-Planck equation

$$\partial_t p_Y = \tfrac{1}{2}\sum_{i,j=1}^{d} \partial_{x_i x_j}\left(\sigma\sigma^\top p_Y\right) - \nabla \cdot (\mu p_Y).$$

# Denoising score matching

**Reverse-time generative SDE**

$$\mathrm{d}X_s = \left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla \log \overleftarrow{p}_{Y_s} - \overleftarrow{\mu}\right)(X_s, s)\mathrm{d}s + \overleftarrow{\sigma}(s)\mathrm{d}B_s, \quad X_0 \sim Y_T.$$

**Idea:**

1. Sample $X_0 \sim \mathcal{N}(0, \mathrm{I})$.
2. Simulate the SDE to obtain samples $X_T \approx Y_0$ (in distribution)

⚠ We need an approximation to the **score** $\nabla \log \overleftarrow{p}_{Y_s}$.

# Denoising score matching

**Reverse-time generative SDE**

$$\mathrm{d}X_s = \left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla \log \overleftarrow{p}_{Y_s} - \overleftarrow{\mu}\right)(X_s, s)\mathrm{d}s + \overleftarrow{\sigma}(s)\mathrm{d}B_s, \quad X_0 \sim Y_T.$$

**Idea:**

1. Sample $X_0 \sim \mathcal{N}(0, \mathrm{I})$.
2. Simulate the SDE to obtain samples $X_T \approx Y_0$ (in distribution)

⚠ We need an approximation to the **score** $\nabla \log \overleftarrow{p}_{Y_s}$.

Having **access to samples** from $Y_0$, one can use the denoising score matching objective

$$\min_\theta \mathbb{E}\left[\left\|\Phi_\theta(Y_\tau, \tau) - \nabla \log p_{Y_\tau | Y_0}(Y_\tau | Y_0)\right\|^2\right], \quad \tau \in \mathrm{Unif}([0, T]),$$

to learn the score $\Phi_\theta(\cdot, s) \approx \nabla \log p_{Y_s}$.

# Sampling from densities via Kolmogorov equations

⚠ We cannot use denoising score matching objective if we only have **access to an (unnormalized) density** $\rho$ with $p_{Y_0} = \rho/\mathcal{Z}$.

**Idea:** The density of our diffusion process satisfies a Kolmogorov backwards PDE.

**Kolmogorov backwards PDE for density**

The scaled reverse-time density $u(x,t) := \mathcal{Z}\overleftarrow{p}_{Y_t}(x)$ solves

$$\partial_t u + \tfrac{1}{2}\operatorname{tr}\left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \operatorname{Hess}_u\right) - \partial_x u \overleftarrow{\mu} - \nabla \cdot \overleftarrow{\mu} u = 0, \quad p(\cdot, T) = \rho.$$

# Sampling from densities via Kolmogorov equations

⚠ We cannot use denoising score matching objective if we only have **access to an (unnormalized) density** $\rho$ with $p_{Y_0} = \rho/\mathcal{Z}$.

**Idea:** The density of our diffusion process satisfies a Kolmogorov backwards PDE.

**Kolmogorov backwards PDE for density**

The scaled reverse-time density $u(x,t) := \mathcal{Z}\overleftarrow{p}_{Y_t}(x)$ solves

$$\partial_t u + \tfrac{1}{2}\operatorname{tr}\left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \operatorname{Hess}_u\right) - \partial_x u \overleftarrow{\mu} - \nabla \cdot \overleftarrow{\mu} u = 0, \quad p(\cdot, T) = \rho.$$

**Proof:** Time-reversal and linearity of the Fokker-Planck equation.

# Sampling from densities via Kolmogorov equations

⚠ We cannot use denoising score matching objective if we only have **access to an (unnormalized) density** $\rho$ with $p_{Y_0} = \rho/\mathcal{Z}$.

**Idea:** The density of our diffusion process satisfies a Kolmogorov backwards PDE.

**Kolmogorov backwards PDE for density**

The scaled reverse-time density $u(x,t) := \mathcal{Z}\overleftarrow{p}_{Y_t}(x)$ solves

$$\partial_t u + \tfrac{1}{2}\operatorname{tr}\left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \operatorname{Hess}_u\right) - \partial_x u \overleftarrow{\mu} - \nabla \cdot \overleftarrow{\mu} u = 0, \quad p(\cdot, T) = \rho.$$

**Proof:** Time-reversal and linearity of the Fokker-Planck equation.

Since we know $\rho$, we can **solve this PDE** using our SDE-based variational formulation and use **automatic differentiation** to obtain an approximation to the score:

$$\Phi_\theta(x,t) \approx u(x,t) = \mathcal{Z}\overleftarrow{p}_{Y_t}(x) \quad \xrightarrow{\text{AD}} \quad \nabla \log \Phi_\theta \approx \nabla \log u = \nabla \log \overleftarrow{p}_{Y_t}.$$

# Sampling from densities via Kolmogorov equations

⚠ We cannot use denoising score matching objective if we only have **access to an (unnormalized) density** $\rho$ with $p_{Y_0} = \rho/\mathcal{Z}$.

**Idea:** The density of our diffusion process satisfies a Kolmogorov backwards PDE.

**Kolmogorov backwards PDE for density**

The scaled reverse-time density $u(x,t) := \mathcal{Z}\overleftarrow{p}_{Y_t}(x)$ solves

$$\partial_t u + \tfrac{1}{2}\operatorname{tr}\left(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \operatorname{Hess}_u\right) - \partial_x u \overleftarrow{\mu} - \nabla \cdot \overleftarrow{\mu} u = 0, \quad p(\cdot, T) = \rho.$$

**Proof:** Time-reversal and linearity of the Fokker-Planck equation.

Since we know $\rho$, we can **solve this PDE** using our SDE-based variational formulation and use **automatic differentiation** to obtain an approximation to the score:

$$\Phi_\theta(x,t) \approx u(x,t) = \mathcal{Z}\overleftarrow{p}_{Y_t}(x) \quad \xrightarrow{\text{AD}} \quad \nabla \log \Phi_\theta \approx \nabla \log u = \nabla \log \overleftarrow{p}_{Y_t}.$$

⚠ Expensive sampling!

# Diffusion-based sampling from densities

**One step further**:

$$\text{Fokker-Planck eq.} \xrightarrow{\text{time-reversal}} \text{Kolmogorov eq.} \xrightarrow{\text{Hopf-Cole transform}} \text{HJB eq.}$$

**Hamilton-Jacobi-Bellman equation for log-density**

The scaled reverse-time log-density $u(x,t) := -\log\left(\mathcal{Z}\overleftarrow{p}_{Y_t}(x)\right)$ solves

$$\partial_t u = -\tfrac{1}{2}\operatorname{tr}(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla^2 u) + \overleftarrow{\mu}\cdot\nabla u - \nabla\cdot\overleftarrow{\mu} + \tfrac{1}{2}\left\|\overleftarrow{\sigma}^\top \nabla u\right\|^2, \quad u(\cdot,T) = -\log(\rho).$$

# Diffusion-based sampling from densities

**One step further**:

$$\text{Fokker-Planck eq.} \xrightarrow{\text{time-reversal}} \text{Kolmogorov eq.} \xrightarrow{\text{Hopf-Cole transform}} \text{HJB eq.}$$

**Hamilton-Jacobi-Bellman equation for log-density**

The scaled reverse-time log-density $u(x,t) := -\log\left(\mathcal{Z}\overleftarrow{p}_{Y_t}(x)\right)$ solves

$$\partial_t u = -\tfrac{1}{2}\operatorname{tr}(\overleftarrow{\sigma}\overleftarrow{\sigma}^\top \nabla^2 u) + \overleftarrow{\mu}\cdot\nabla u - \nabla\cdot\overleftarrow{\mu} + \tfrac{1}{2}\left\|\overleftarrow{\sigma}^\top\nabla u\right\|^2, \quad u(\cdot,T) = -\log(\rho).$$

We can use the **verification theorem** from stochastic optimal control to establish

$$\underbrace{\overleftarrow{\sigma}^\top \nabla \log\left(\overleftarrow{p}_Y\right)}_{\text{scaled score}} = \arg\min_{v\in\mathcal{V}} \mathbb{E}\left[\int_0^T \underbrace{\left(\nabla\cdot\overleftarrow{\mu} + \tfrac{1}{2}\|v\|^2\right)(X_s^v, s)}_{\text{running costs}}\,\mathrm{d}s \underbrace{-\log\rho(X_T^v)}_{\text{terminal costs}}\right].$$

where the controlled SDE $X^v$ is given by

$$\mathrm{d}X_s^v = \left(\overleftarrow{\sigma}v - \overleftarrow{\mu}\right)(X_s^v, s)\mathrm{d}s + \overleftarrow{\sigma}(s)\mathrm{d}B_s, \quad X_0^v \sim Y_T.$$

# Diffusion-based sampling from densities

**Objective**

$$\underbrace{\breve{\sigma}^\top \nabla \log\left(\breve{p}_Y\right)}_{\text{scaled score}} = \arg\min_{v \in \mathcal{V}} \mathbb{E}\left[\int_0^T \left(\nabla \cdot \breve{\mu} + \tfrac{1}{2}\|v\|^2\right)(X_s^v, s)\,\mathrm{d}s - \log \rho(X_T^v)\right].$$

1. Parametrize $v$ by a NN $\Phi_\theta$ and optimize the objective (using the adjoint method (Kidger et al., 2021)).
2. Sample $X_0 \sim \mathcal{N}(0, \mathrm{I})$ and simulate the SDE to obtain samples from $X_T^{\Phi_\theta^*} \approx Y_0$ (in distribution).

# Diffusion-based sampling from densities

## Objective

$$\underbrace{\tilde{\sigma}^\top \nabla \log\left(\tilde{p}_Y\right)}_{\text{scaled score}} = \arg\min_{v \in \mathcal{V}} \mathbb{E}\left[\int_0^T \left(\nabla \cdot \tilde{\mu} + \tfrac{1}{2}\|v\|^2\right)(X_s^v, s)\,\mathrm{d}s - \log \rho(X_T^v)\right].$$

1. Parametrize $v$ by a NN $\Phi_\theta$ and optimize the objective (using the adjoint method (Kidger et al., 2021)).
2. Sample $X_0 \sim \mathcal{N}(0, \mathrm{I})$ and simulate the SDE to obtain samples from $X_T^{\Phi_\theta^*} \approx Y_0$ (in distribution).

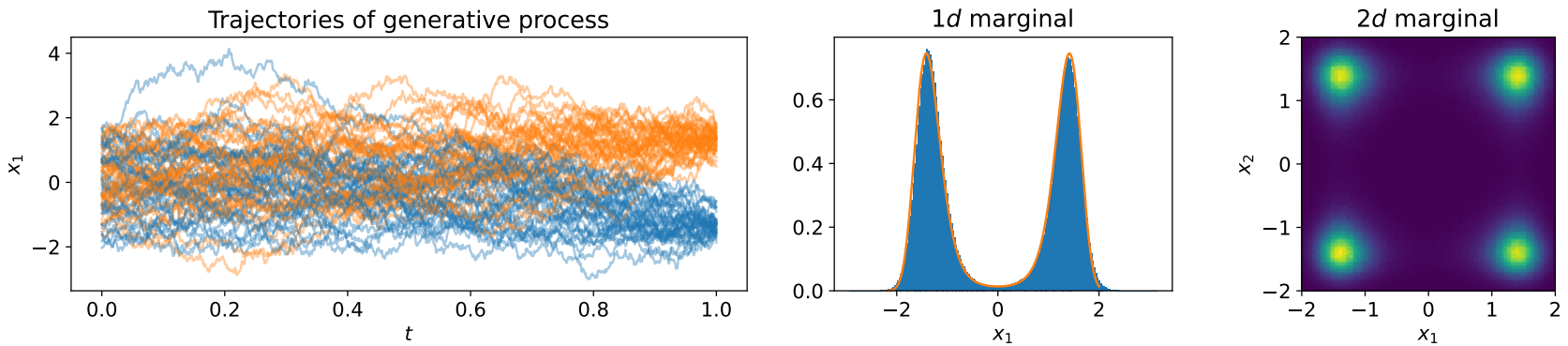



**Figure 12:** Sampling from $\rho(x) = \exp\left(-\sum_{i=1}^{5}(x_i^2 - 2)^2 - \frac{1}{2}\sum_{i=6}^{50} x_i^2\right)$ (Berner, Richter, et al., 2022).

⚠ This is related to methods in stochastic optimal control (Dai Pra, 1991; Nüsken & Richter, 2021; Zhang & Chen, 2022).

# Thank you for your attention!

(it was all I needed)

Homepage: jberner.info

---


Berner, J., Elbrächter, D. M., & Grohs, P. (2019). How degenerate is the parametrization of neural networks with the ReLU activation function? *Advances in Neural Information Processing Systems*, *32*.

Berner, J., Dablander, M., & Grohs, P. (2020). Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 16615–16627.

Berner, J., Grohs, P., & Jentzen, A. (2020). Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, *2*(3), 631–657.

Berner, J., Grohs, P., Kutyniok, G., & Petersen, P. (2022). The modern mathematics of deep learning. In *Mathematical aspects of deep learning*. Cambridge University Press.

Berner, J., Richter, L., & Ullrich, K. (2022). An optimal control perspective on diffusion-based generative modeling [To appear].

Richter, L., & Berner, J. (2022). Robust SDE-based variational formulations for solving linear PDEs via deep learning. *International Conference on Machine Learning*, 18649–18666.

# References i


Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, *12*(3), 313–326.

Dai Pra, P. (1991). A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, *23*(1), 313–329.

Guo, X., Li, W., & Iorio, F. (2016). Convolutional neural networks for steady flow approximation. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 481–490.

Baldi, P. (2017). *Stochastic calculus: An introduction through theory and exercises*. Springer International Publishing.

E, W., Han, J., & Jentzen, A. (2017a). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, *5*(4), 349–380.

E, W., Han, J., & Jentzen, A. (2017b). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, *5*(4), 349–380.

Jentzen, A., Salimova, D., & Welti, T. (2018). A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*.

Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, *375*, 1339–1364.

Solernou, A., Hanson, B. S., Richardson, R. A., Welch, R., Read, D. J., Harlen, O. G., & Harris, S. A. (2018). Fluctuating finite element analysis (ffea): A continuum mechanics software tool for mesoscale simulation of biomolecules. *PLoS computational biology*, *14*(3), e1005897.

Berner, J., Elbrächter, D. M., & Grohs, P. (2019). How degenerate is the parametrization of neural networks with the ReLU activation function? *Advances in Neural Information Processing Systems*, *32*.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., & Kaushik, S. (2019). Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, *64*(2), 525–545.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707.

Berner, J., Dablander, M., & Grohs, P. (2020). Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 16615–16627.

Berner, J., Grohs, P., & Jentzen, A. (2020). Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, *2*(3), 631–657.

Bhattacharya, K., Hosseini, B., Kovachki, N. B., & Stuart, A. M. (2020). Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*.

# References ii


Chen, F., Huang, J., Wang, C., & Yang, H. (2020). Friedrichs learning: Weak solutions of partial differential equations via deep learning. *arXiv preprint arXiv:2012.08023*.

Grohs, P., & Herrmann, L. (2020). Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions. *arXiv preprint arXiv:2007.05384*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Reisinger, C., & Zhang, Y. (2020). Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. *Analysis and Applications*, *18*(06), 951–999.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., & Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1457–1466.

Beck, C., Becker, S., Grohs, P., Jaafari, N., & Jentzen, A. (2021). Solving the Kolmogorov PDE by means of deep learning. *Journal of Scientific Computing*, *88*(3), 1–28.

Kidger, P., Foster, J., Li, X. C., & Lyons, T. (2021). Efficient and accurate gradients for neural SDEs. *Advances in Neural Information Processing Systems*, *34*, 18747–18761.

Kingma, D., Salimans, T., Poole, B., & Ho, J. (2021). Variational diffusion models. *Advances in Neural Information Processing Systems*, *34*, 21696–21707.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*.

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, *3*(3), 218–229.

Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *International Conference on Machine Learning*, 8162–8171.

Nüsken, N., & Richter, L. (2021). Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: Perspectives from the theory of controlled diffusions and measures on path space. *Partial Differential Equations and Applications*, *2*(4), 1–48.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models.

Berner, J., Grohs, P., Kutyniok, G., & Petersen, P. (2022). The modern mathematics of deep learning. In *Mathematical aspects of deep learning*. Cambridge University Press.

Berner, J., Richter, L., & Ullrich, K. (2022). An optimal control perspective on diffusion-based generative modeling [To appear].

Brandstetter, J., Worrall, D., & Welling, M. (2022). Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*.

# References iii


Corso, G., Stärk, H., Jing, B., Barzilay, R., & Jaakkola, T. (2022). Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*.

De Hoop, M., Huang, D. Z., Qian, E., & Stuart, A. M. (2022). The cost-accuracy trade-off in operator learning with neural networks. *arXiv preprint arXiv:2203.13181*.

Gupta, J. K., & Brandstetter, J. (2022). Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*.

Hoogeboom, E., Satorras, V. G., Vignac, C., & Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. *International Conference on Machine Learning*, 8867–8887.

Jing, B., Corso, G., Chang, J., Barzilay, R., & Jaakkola, T. (2022). Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*.

Qiao, Z., Nie, W., Vahdat, A., Miller III, T. F., & Anandkumar, A. (2022). Dynamic-backbone protein-ligand structure prediction with multiscale generative diffusion models. *arXiv preprint arXiv:2209.15171*.

Richter, L., & Berner, J. (2022). Robust SDE-based variational formulations for solving linear PDEs via deep learning. *International Conference on Machine Learning*, 18649–18666.

Trippe, B. L., Yim, J., Tischer, D., Broderick, T., Baker, D., Barzilay, R., & Jaakkola, T. (2022). Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., & Tang, J. (2022). Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*.

Zhang, Q., & Chen, Y. (2022). Path integral sampler: A stochastic control approach for sampling. *International Conference on Learning Representations*.