The 38th Annual AAAI Conference on Artificial Intelligence

Special track on Safe, Robust and Responsible Artificial Intelligence (SRRAI)

Physics-informed Representation and Learning: Control and Risk Quantification

Zhuoyuan Wang[1], Reece Keller[1], Xiyu Deng[1], Kenta Hoshino[2], Takashi Tanaka[3], Yorie Nakahira[1]

[1]Carnegie Mellon University
[2]Kyoto University
[3]University of Texas at Austin

Carnegie Mellon University

Motivation

Control and Risk Quantification

Optimal and Safe Control of high-dimensional stochastic systems

Source: Andy Dean/stock.adobe.com.

Performance requirement

(e.g. certain formation)

→ Optimal control

Safety requirement

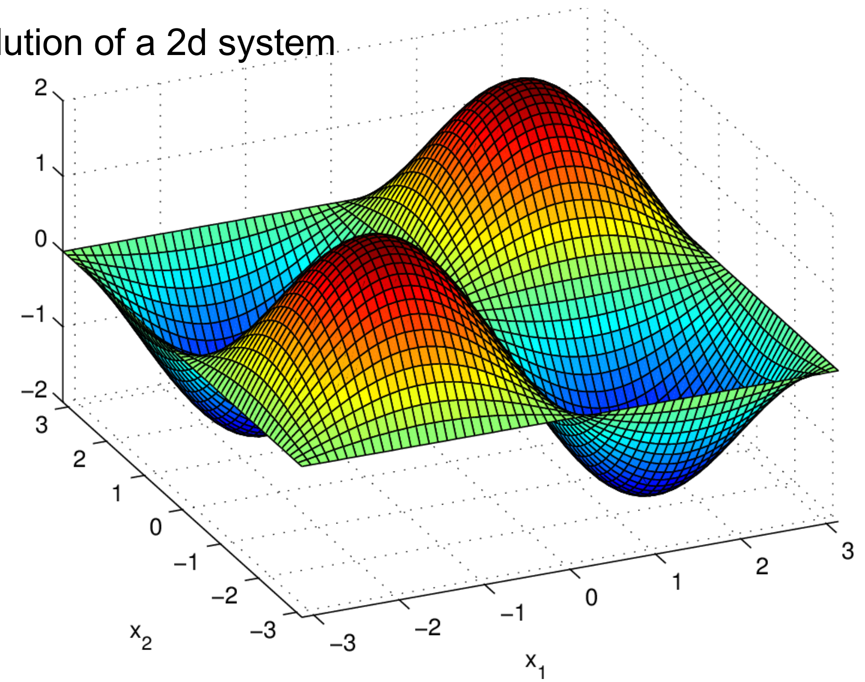
(e.g. no collision)

→ Safe control

Challenges

- **Need to estimate value function and safety probability**
- **Scale exponentially with system dimension**
- **Solving corresponding high-dimensional partial differential equations (PDEs) is hard**

HJB solution of a 2d system



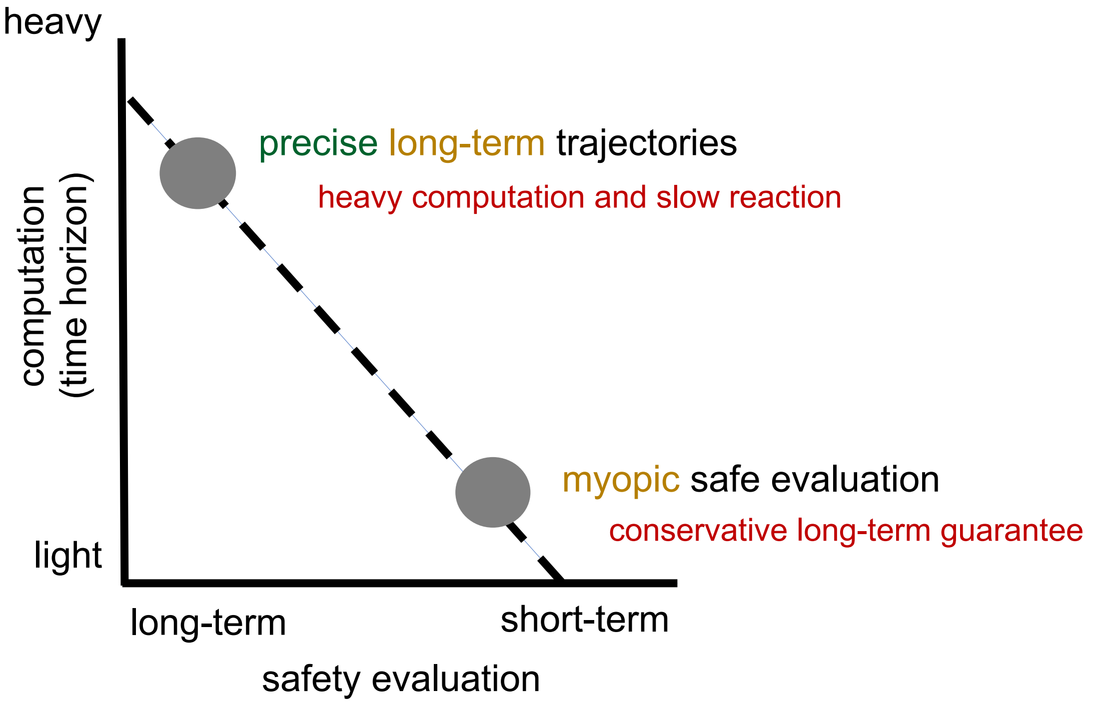
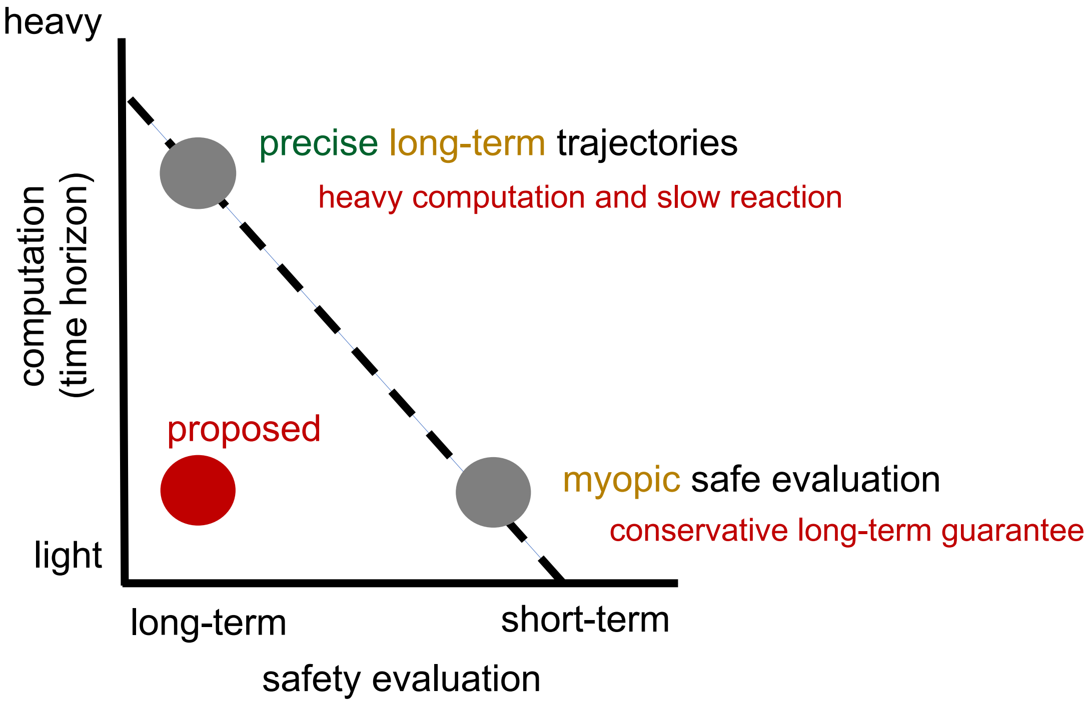


Image source: Greif, Constantin. "Numerical methods for hamilton-jacobi-bellman equations." (2017).

Challenges

- **Need to estimate value function and safety probability**
- **Scale exponentially with system dimension**
- **Solving corresponding high-dimensional partial differential equations (PDEs) is hard**

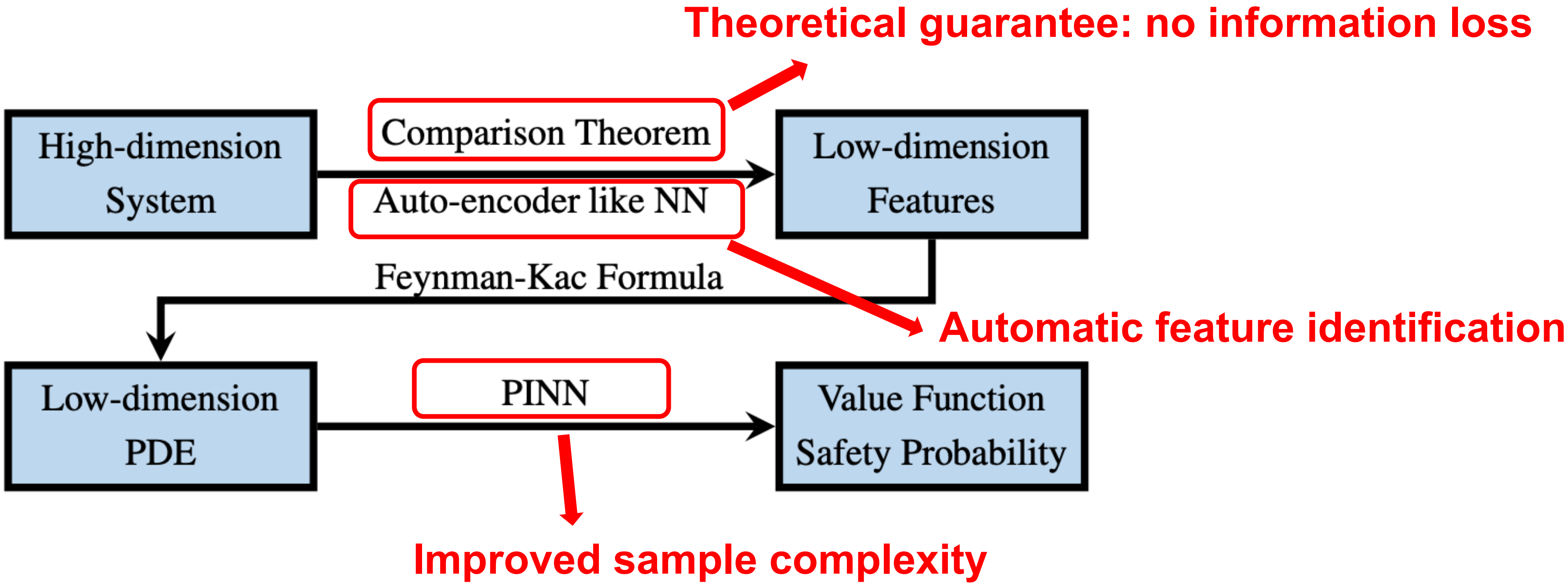
Can we do this efficiently?



Proposed Method: Overview

Goal: **Efficient estimation of optimal value function / safety probability of high-dimensional stochastic systems in one unified framework**

Overall diagram:



Carnegie Mellon University

Problem Statement (Optimal Control)

System description: $$dx_t = f\left(x_t\right) dt + \sigma\left(x_t\right)\left(u_t dt + dw_t\right) \qquad x_t \in \mathcal{X} \subset \mathbb{R}^n$$

Stochastic differential equation (SDE)

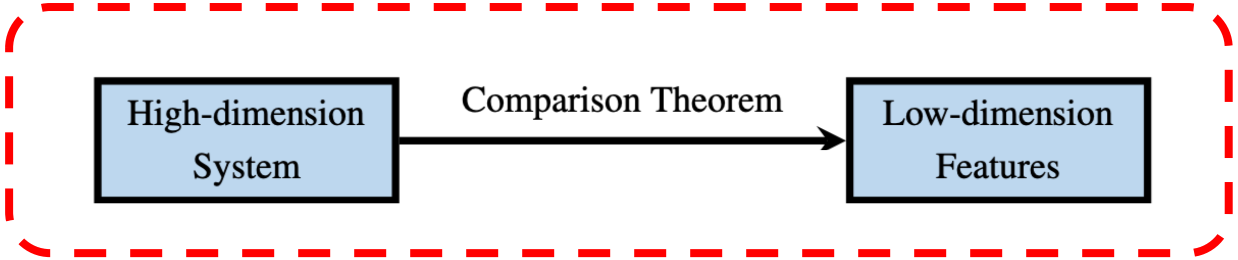
Running-cost function: $$w(x_t, u_t) = c(x_t) + \frac{1}{2}\|u_t\|^2$$

Quadratic cost on control

Optimal value function: $$V(x,t) := \min_u \mathbb{E}^P\left[\int_t^T w(x_\tau, u_\tau) d\tau + c\left(x_T\right) \mid x_t = x\right]$$

Minimum expected cumulative cost

Low-dimensional Feature



One-dimensional feature: smooth function $p : \mathbb{R}^n \to \mathbb{R}$

Definition of operation:

$$\mathcal{A}^U(\cdot)(x) = \frac{\partial(\cdot)}{\partial x}(x) f(x) + \frac{\partial(\cdot)}{\partial x}(x)\sigma(x)U + \frac{1}{2}\mathrm{Tr}\left(\frac{\partial^2(\cdot)}{\partial x^2}(x)\sigma(x)\sigma(x)^\top\right),$$

Definition of variables:

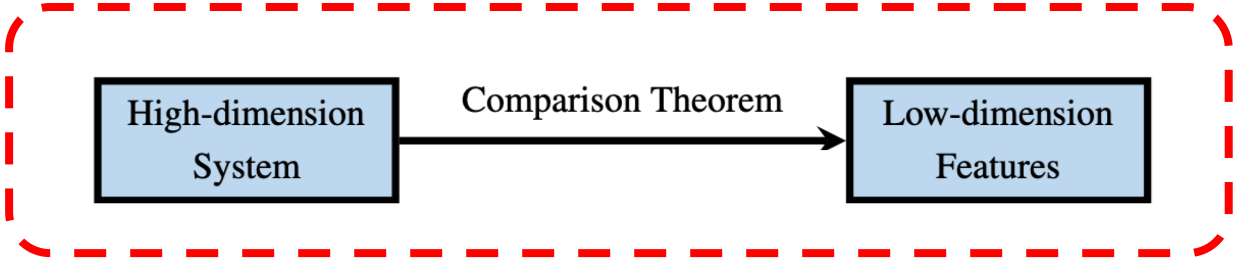
$$a(x) = \sum_{i,j,k} \sigma_k^i(x)\sigma_k^j(x)\frac{\partial p}{\partial x_i}(x)\frac{\partial p}{\partial x_j}(x),\ b(x) = \frac{\mathcal{A}^U p(x)}{a(x)}$$

Upper bounds and lower bounds for the fixed feature:

$$a^+(\xi) = \sup_{x:p(x)=\xi} a(x), \quad a^-(\xi) = \inf_{x:p(x)=\xi} a(x)$$

$$b^+(\xi) = \sup_{x:p(x)=\xi} b(x), \quad b^-(\xi) = \inf_{x:p(x)=\xi} b(x)$$

One-dimensional Process



Upper bounds and lower bounds for the fixed feature:

$$a^+(\xi) = \sup_{x:p(x)=\xi} a(x), \quad a^-(\xi) = \inf_{x:p(x)=\xi} a(x)$$

$$b^+(\xi) = \sup_{x:p(x)=\xi} b(x), \quad b^-(\xi) = \inf_{x:p(x)=\xi} b(x)$$

Matched upper and lower bounds → **1-dimensional stochastic process representation**

No information loss

Assumption 2. *We assume that the feature function* $p(x)$ *satisfies that* $a^+(\xi) = a^-(\xi) = \alpha(\xi)$ *and* $b^+(\xi) = b^-(\xi) = \beta(\xi)$, $\forall \xi \in I$.

Assumption 3. *The functions* $\alpha(\xi)$ *and* $\beta(\xi)$ *are globally Lipschitz continuous in* $\xi \in I \subset \mathbb{R}$. *Moreover,* $a(x) > 0$, $\forall x \in \mathbb{R}^n$.

Theorem 4. *Given Assumptions 2 and 3 hold,* $p(x_t)$ *with* x_t *being sampled from system* (1) *is characterized by the following stochastic process*

$$d\xi_t = \alpha(\xi_t)\,\beta(\xi_t)\,dt + \sqrt{\alpha(\xi_t)}\,d\tilde{B}_t, \tag{14}$$

with $\xi_0 = p(x_0)$, *and* $\tilde{B}_t$ *being a one-dimensional standard Wiener process.*

Stochastic Process to PDE

Let $$V(x,t) = -\log \varphi(x,t).$$

(logarithmic transformation of the value function)

$$\xi = p(x) = c(x)$$

(running-cost function as feature)

Feynman-Kac formula

1-dimensional stochastic process representation → **2-dimensional PDE**

$$\varphi(x,t) = \mathbb{E}\left[\exp\left(-\int_t^T \xi_\tau d\tau - \xi_T\right) \mid \xi_t = \xi\right]$$

Path integral control

$$W_\varphi(\xi,t) := \frac{\partial \varphi}{\partial t}(\xi,t) + \alpha(\xi,t)\beta(\xi,t)\frac{\partial \varphi}{\partial \xi}(\xi,t) + \frac{1}{2}\alpha(\xi,t)\frac{\partial^2 \varphi}{\partial \xi^2}(\xi,t) - \xi\varphi(\xi,t) = 0,$$

with initial (terminal) condition

$$\varphi(\xi,T) = \exp(-\frac{1}{2}\xi).$$

Multi-dimensional Feature

Extension to multi-dimensional features:

Let $p(x) = [p_1(x), p_2(x), \cdots, p_k(x)]^\top$

Assumption 6. *For value function estimation, we assume the running-cost can be represented by the following function*

$$c(x) = r(\xi) = r\left(p_1(x), p_2(x), \cdots, p_k(x)\right)$$

where $r : \mathbb{R}^k \to \mathbb{R}$ is a continuous function.

Low-dimensional stochastic process representation → **Low-dimensional PDE**

$$d\xi_t^{(i)} = \alpha_i\left(\xi_t^{(i)}\right)\beta_i\left(\xi_t^{(i)}\right)dt + \sqrt{\alpha_i\left(\xi_t^{(i)}\right)}d\tilde{B}_t^{(i)}$$

for $i = 1, 2, \cdots, k$ that characterize $p_i(x)$, where $\tilde{B}_t^{(i)}$ is one-dimensional standard Wiener processes.

Comparison theorem

Path integral control

Then

$$\varphi(x,t) = \mathbb{E}\left[\exp\left(-\int_t^T r(\xi_\tau)d\tau - r(\xi_T)\right) \mid \xi_t = \xi\right],$$

$$r\varphi - \frac{\partial\varphi}{\partial t} - \mathcal{G}_t\varphi = 0, \quad \text{on } \mathbb{R}^k \times [0,T)$$

$$\varphi(\cdot, T) = \exp(-r(\cdot)), \quad \text{on } \mathbb{R}^k,$$

Feynman-Kac formula

Example

System dynamics:

$$dx_1 = (x_1 + x_3)\, dt + \sigma_1 (u_1 dt + dW_1)$$
$$dx_2 = (x_2 - x_3)\, dt + \sigma_2 (u_2 + dW_2)$$
$$dx_3 = x_3\, dt + \sigma_3 (u_3 + dW_3)$$

Running-cost function:

$$c(x) = \frac{1}{2}(x_1 + x_2)^2 + \frac{1}{2}x_3^2.$$

Features:

$$\xi_1 = p_1(x) = x_1 + x_2, \quad \xi_2 = p_2(x) = x_3$$

Matched upper and lower bounds:

$$\alpha_1(\xi_1) = 2,\ \alpha_2(\xi_2) = 1; \quad \beta_1(\xi_1) = \frac{\xi_1}{2},\ \beta_2(\xi_2) = \xi_2$$

Reconstruction of running-cost function:

$$r(\xi) = \frac{1}{2}\xi_1^2 + \frac{1}{2}\xi_2^2$$

Example

PDE whose solution will give value function:

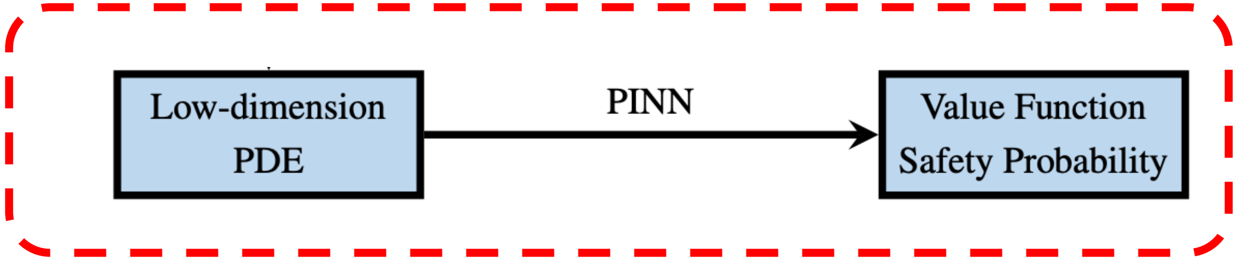
$$0 = r(\xi)\mu - \frac{\partial \mu}{\partial t} - \alpha(\xi)\beta(\xi)\frac{\partial \mu}{\partial \xi} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{a}(\xi)\frac{\partial^2 \mu}{\partial \xi^2}\right)$$

$$= (\frac{1}{2}\xi_1^2 + \frac{1}{2}\xi_2^2)\mu - \frac{\partial \mu}{\partial t} - \xi_1\frac{\partial \mu}{\partial \xi_1} - \xi_2\frac{\partial \mu}{\partial \xi_2} - \frac{\partial^2 \mu}{\partial \xi_1^2} - \frac{1}{2}\frac{\partial^2 \mu}{\partial \xi_2^2}$$

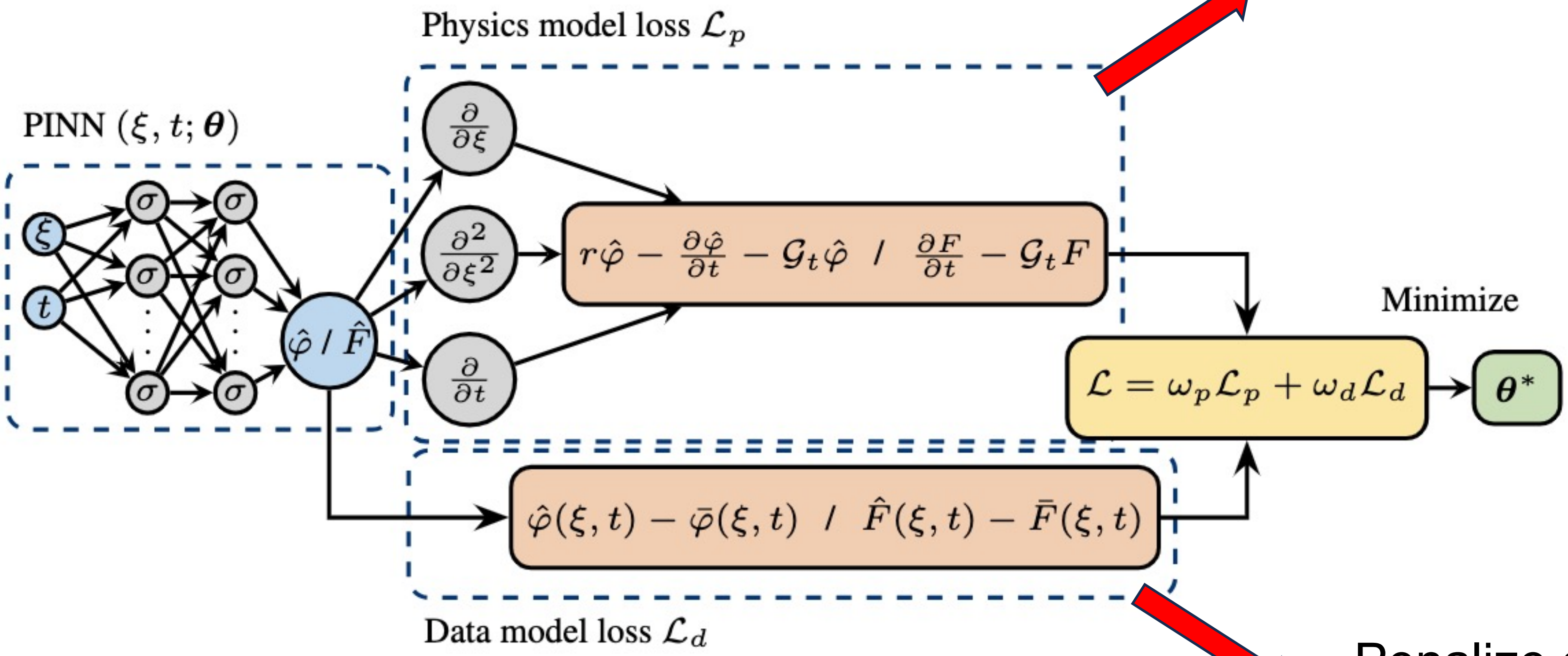
$$\mu(\xi, T) = \exp(-\frac{1}{2}\xi_1^2 - \frac{1}{2}\xi_2^2)$$

3D system -> 2D feature

Physics-informed Learning



Solve PDE with PINN

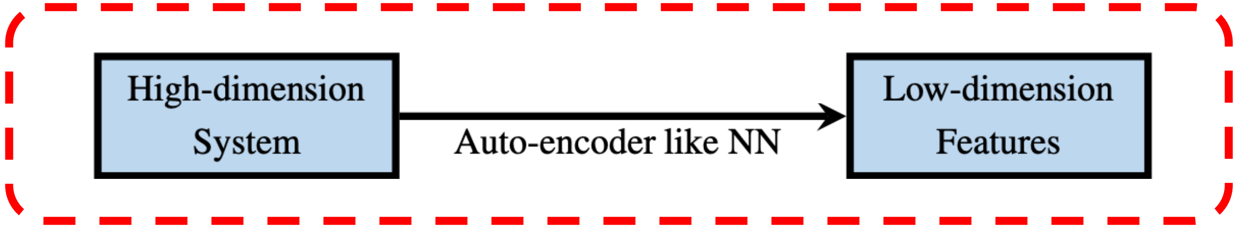


Penalize dissatisfaction of the PDE

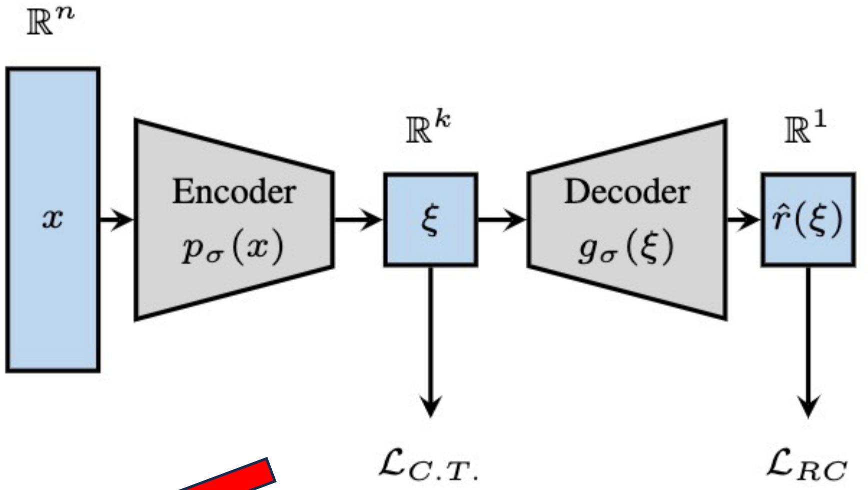
Penalize difference between prediction and training data

Lu, Lu, et al. "DeepXDE: A deep learning library for solving differential equations." *SIAM review* 63.1 (2021): 208-228.

Automatic Feature Finding



Find features using auto-encoder like neural network



Comparison Theorem loss

Penalize difference between upper and lower bound

$$a^+(\xi) = \sup_{x:p(x)=\xi} a(x), \quad a^-(\xi) = \inf_{x:p(x)=\xi} a(x)$$

$$b^+(\xi) = \sup_{x:p(x)=\xi} b(x), \quad b^-(\xi) = \inf_{x:p(x)=\xi} b(x)$$

Reconstruction loss

Penalize difference between reconstructed and actual running cost function

$$c(x) = r(\xi) = r\left(p_1(x), p_2(x), \cdots, p_k(x)\right)$$

where $r : \mathbb{R}^k \to \mathbb{R}$ *is a continuous function.*

Experiments

Smoother and more accurate prediction

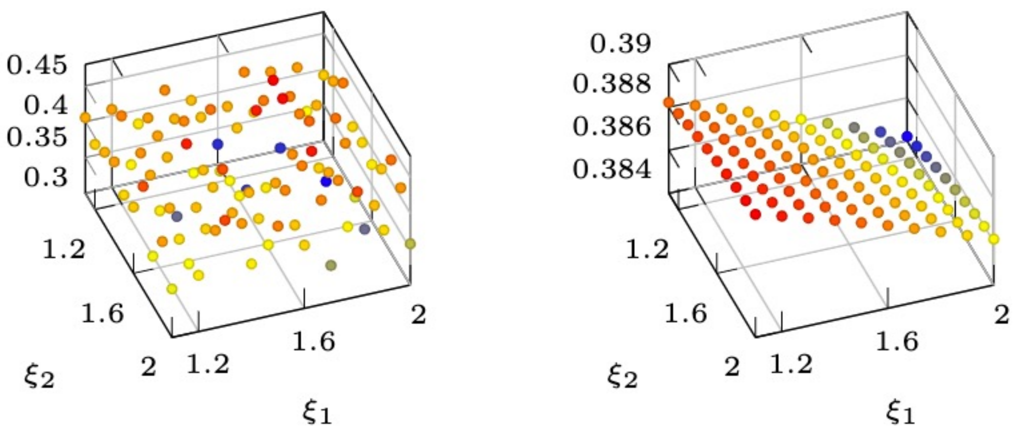


Figure 3: Estimation of the exponential of value function at $t = 0.5$ for a 1000-dimensional system by path integral MC (left), and by the proposed method (right).

Better sample complexity

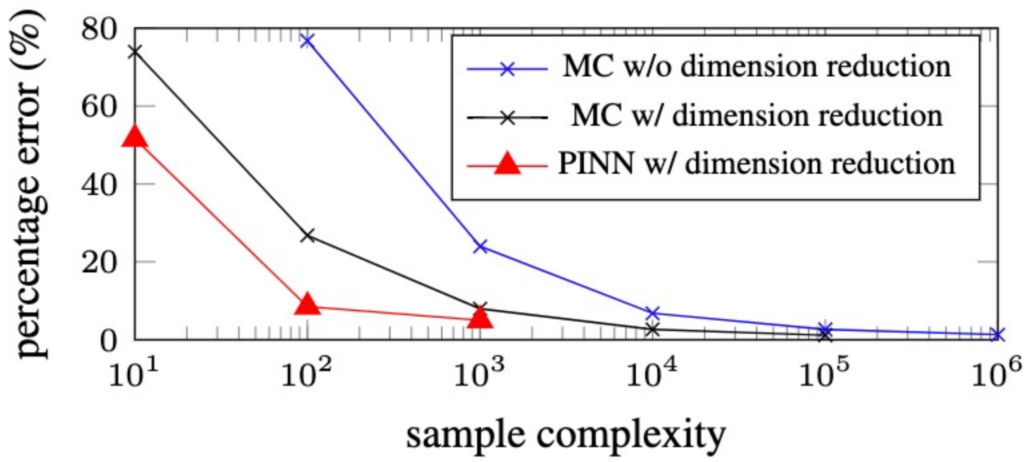


Figure 4: Percentage error of the estimated value function with path integral MC.

Accurate identification of features with the auto-encoder NN

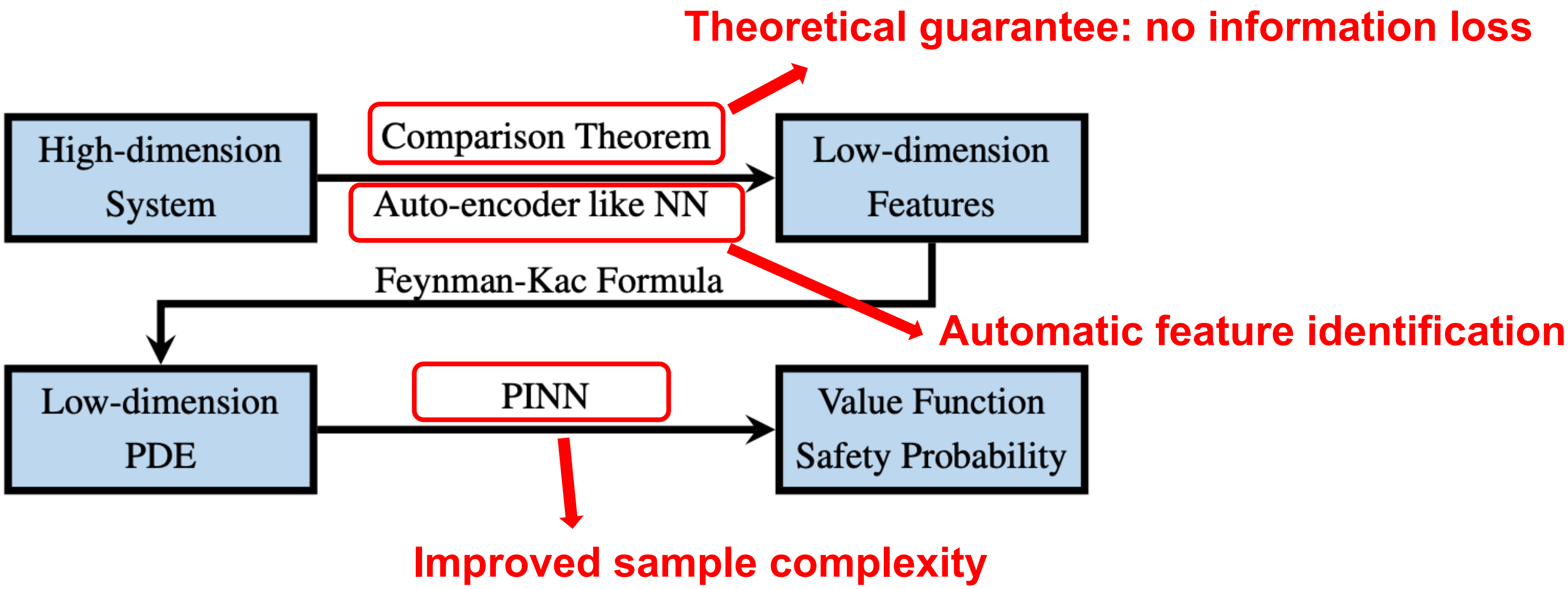
Network Architecture	[16, 64, 128, 64, 16]	[64, 64, 128, 64, 64]	[64, 128, 128, 128, 64]
Representation Error (%)	0.007	0.011	0.009
Cost Reconstruction Error (%)	0.120	0.099	0.129

Table 5: Autoencoder-like network architecture for $n_{\text{layer}} = 5$

Conclusion

Goal: **Efficient estimation of optimal value function / safety probability of high-dimensional stochastic systems in one unified framework**

Overall diagram:



Carnegie Mellon University

Thanks for Listening!

arXiv paper:

Contact: zhuoyuaw@andrew.cmu.edu