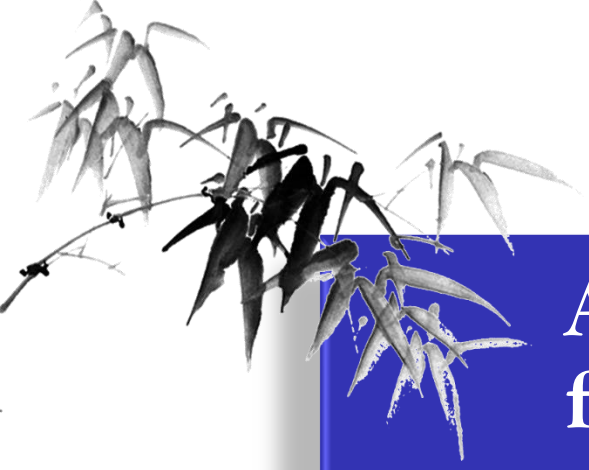


Active Learning Based Error Sampling for High-dimensional Nonlinear PDEs

Wenhan Gao

Stony Brook University

TTU REU Summer 2021

July 30th, 2021

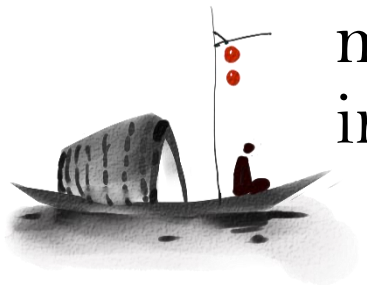
Mentors:

Chunmei Wang

Haizhao Yang (Purdue University)

Introduction and Motivation

- Recent developments in computing capability and large storage media has made Deep Learning the fastest growing field in Artificial Intelligence. Deep Learning has had extraordinary successes in many fields including computer vision, natural language processing, etc..
- Therefore, many recent studies have contemplated deep-learning-based approaches for solving high-dimensional PDEs. But the convergence is usually slow and not guaranteed. In this work, the main objective is to address this issue by sampling the most informative training examples.

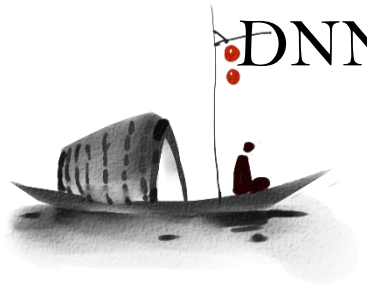


Neural Networks

- A feedforward neural network is given by a compositional function:

$$\phi((\boldsymbol{x};\boldsymbol{\theta}) = h_L \circ h_{L-1} \circ \ldots \circ h_1 \circ h_0(\boldsymbol{x})$$

- Each h consists of linear transformation $wx + b$ and non-linear activation functions such as Tanh, Sigmoid, ReLU.
- Universal Approximation Theorems indicate that deep neural networks, DNNs, can approximate most functions in the Sobolev space.



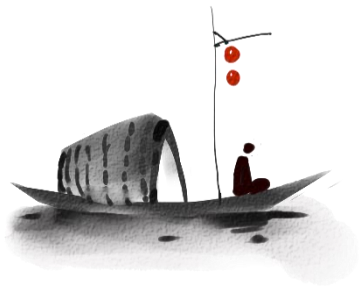
The Residual Model

- Consider the following general form of PDE for u(x):

$$\begin{cases} \mathcal{D}u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ \mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}) & \text{on } \partial\Omega \end{cases}$$

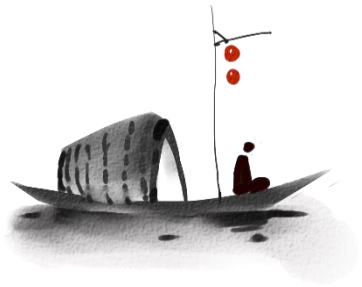
- The PDE is solved by the following optimization constrant:

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) :&= \arg\min_{\boldsymbol{\theta}} \|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\|_2^2 + \lambda\|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\|_2^2 \\ &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \lambda\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right] \\ &\approx \arg\min_{\boldsymbol{\theta}} \frac{1}{N_1}\sum_{i=1}^{N_1} |\mathcal{D}\phi\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right) - f\left(\boldsymbol{x}_i\right)|^2 + \frac{\lambda}{N_2}\sum_{j=1}^{N_2} |\mathcal{B}\phi\left(\boldsymbol{x}_j;\boldsymbol{\theta}\right) - g\left(\boldsymbol{x}_j\right)|^2 \end{aligned}$$



Error Sampling

- In the residual model, a certain number of allocation points are sampled in each epoch for the network to learn, and error sampling is proposed in this work to preferentially choose allocation points with larger residual errors. Intuitively, one can think of high residual error as a proxy of the model being wrong to a greater extent.

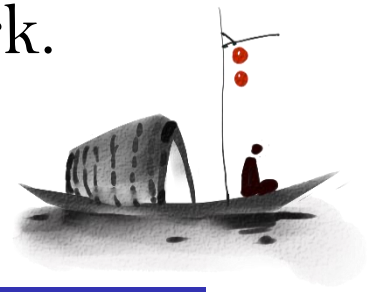


Error Sampling Importance Distribution

- The fundamental methodology of error sampling is to choose from a biased importance distribution $q(x) \propto \mathcal{R}^p_{abs}(x)$that attaches higher priority to important volumes/regions:

$$q(\boldsymbol{x}) = \frac{\mathcal{R}^p_{abs}(\boldsymbol{x})}{NC}$$

,where $\mathcal{R}^p_{abs}(x)$ is the absolute residual error and $NC = \int_\Omega \mathcal{R}^p_{abs}(x)dx$ is the unknown normalzing constant. A self-normalized sampling algorithm to simulate observations from $q(x)$ is proposed and used in this work.



Self-normalized Sampling

Algorithm 3.3: Self-normalized Sampling

Result: N_1 points in Ω for training

Require: PDE (2.1); the current solution net $\phi(\boldsymbol{\theta})$ (2.2)

1 Generate N_1 uniformly distributed points $\{\boldsymbol{x}_i\}_{i=1}^{N_1} \subset \Omega$; denote by $\boldsymbol{X}$;

2 RE_array = $\mathcal{R}_{abs}^{p}(\boldsymbol{X}) = |\mathcal{D}\phi(\boldsymbol{X}) - f(\boldsymbol{X})|^p$;

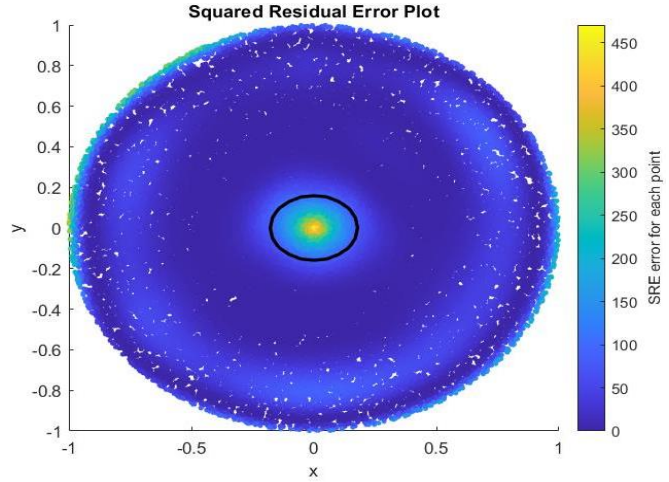
3 SRE = sum(RE_array) ;

4 probability_array = $\frac{\text{RE\_array}}{\text{SRE}}$;

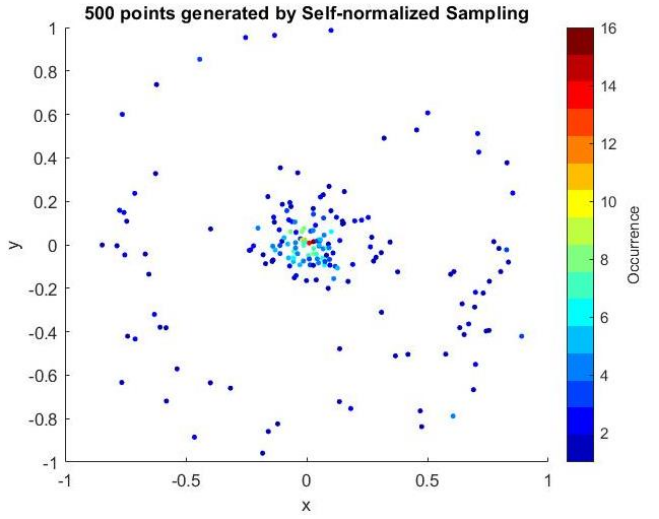
5 Generate N_1 points, X_training following discrete probability density function p(RE_array[i]) = probability_array[i] ;

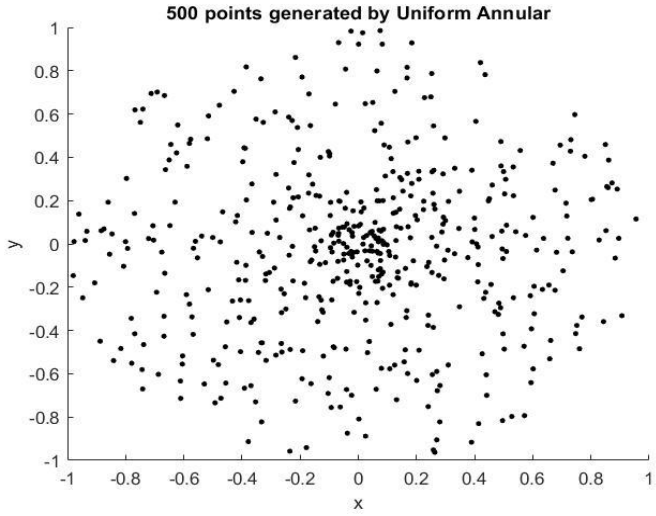
6 **return** X_training ;

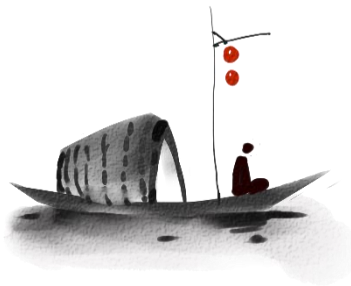
Illustration of Sampling



Approach	Number of Points inside Ellipse
No Error Sampling	138
Error Sampling	348







Numerical Examples

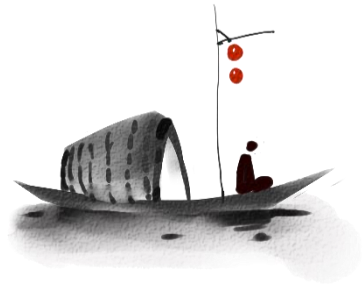
- First, elliptic equation(5.1):

$$-\nabla \cdot \left((1 + \frac{1}{2}|\boldsymbol{x}|^2)\nabla u \right) + (\nabla u)^2 = f(\boldsymbol{x}), \quad \text{in } \Omega := \{\boldsymbol{x} : |\boldsymbol{x}| < 1\}$$

$$u = g(\boldsymbol{x}), \quad \text{on } \partial\Omega,$$

, where $g(x) = 0$ and $f(x)$ is specified appropriately so that the exact solution is:

$$u(\boldsymbol{x}) = sin(\frac{\pi}{2}(1 - |\boldsymbol{x}|)^{2.5})$$



Numerical Examples

Dimension		Error Sampling	Basic Model
10 D	ℓ_2 error	9.121454e-03	2.482386e-02
	max modulus error	3.369123e-02	1.239435e-01
	Running Time in Seconds	8844.328335	7528.474081
20 D	ℓ_2 error	3.193670e-02	7.046980e-02
	max modulus error	1.083703e-01	2.838619e-01
	Running Time in Seconds	12554.326195	10129.247696
100 D	ℓ_2 error	1.142189e-01	5.174087e-01
	max modulus error	2.919715e-01	1.755006e+00
	Running Time in Seconds	52927.455038	40356.392521

Accuracy Assessment

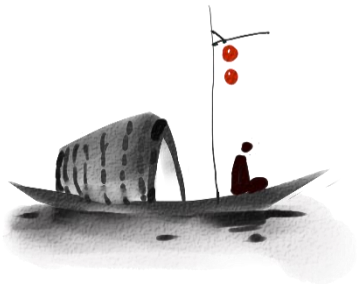
$\{\boldsymbol{x}_i^t\}_{i=1}^{10000} \subset \Omega$, are 10,000 testing points different from training points.

- overall L2 error:

$$e_{\ell^2}^{overall}(\boldsymbol{\theta}) := \frac{\left(\sum_{i=1}^{10000} \left|\phi\left(\boldsymbol{x}_i^t;\boldsymbol{\theta}\right) - u\left(\boldsymbol{x}_i^t\right)\right|^2\right)^{\frac{1}{2}}}{\left(\sum_{i=1}^{10000} \left|u\left(\boldsymbol{x}_i^t\right)\right|^2\right)^{\frac{1}{2}}}$$

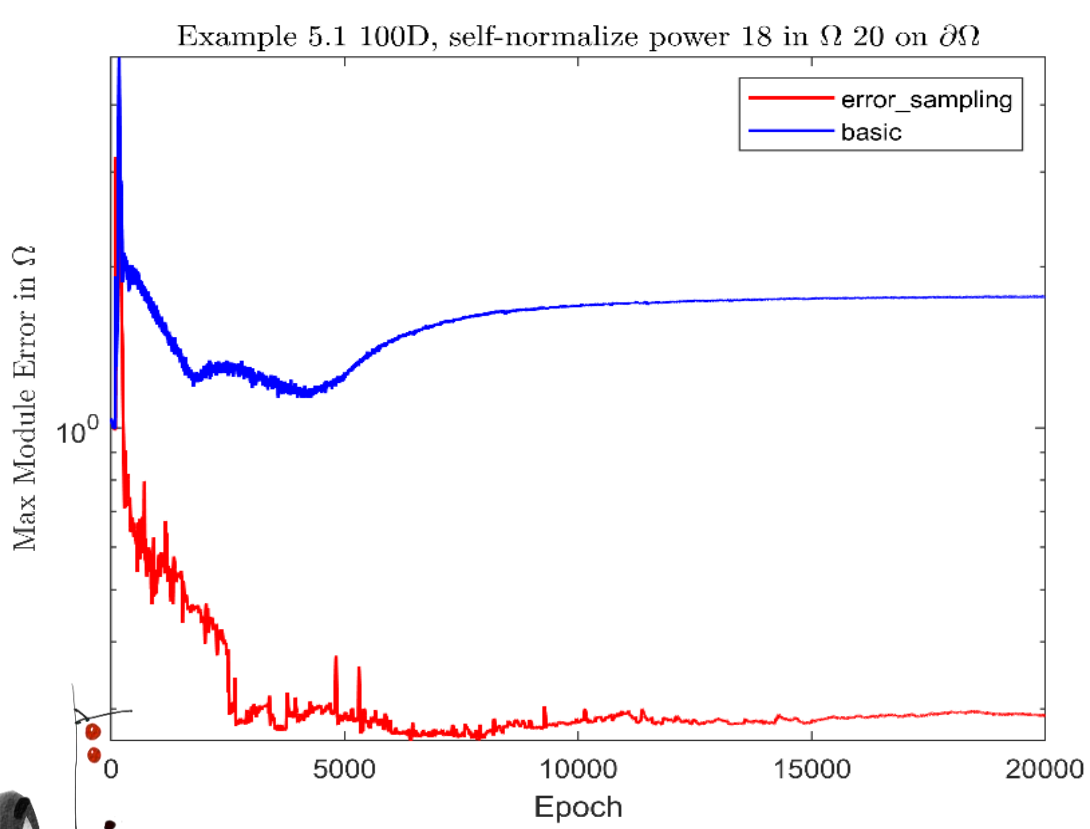
- maximum L1 modulus error:

$$e_{modulus}^{max}(\boldsymbol{\theta}) := \frac{max\left(\left|\phi\left(\boldsymbol{x}_i^t;\boldsymbol{\theta}\right) - u\left(\boldsymbol{x}_i^t\right)\right|\right)_{i=1}^{10000}}{max\left(\left|u\left(\boldsymbol{x}_i^t\right)\right|\right)_{i=1}^{10000}}$$

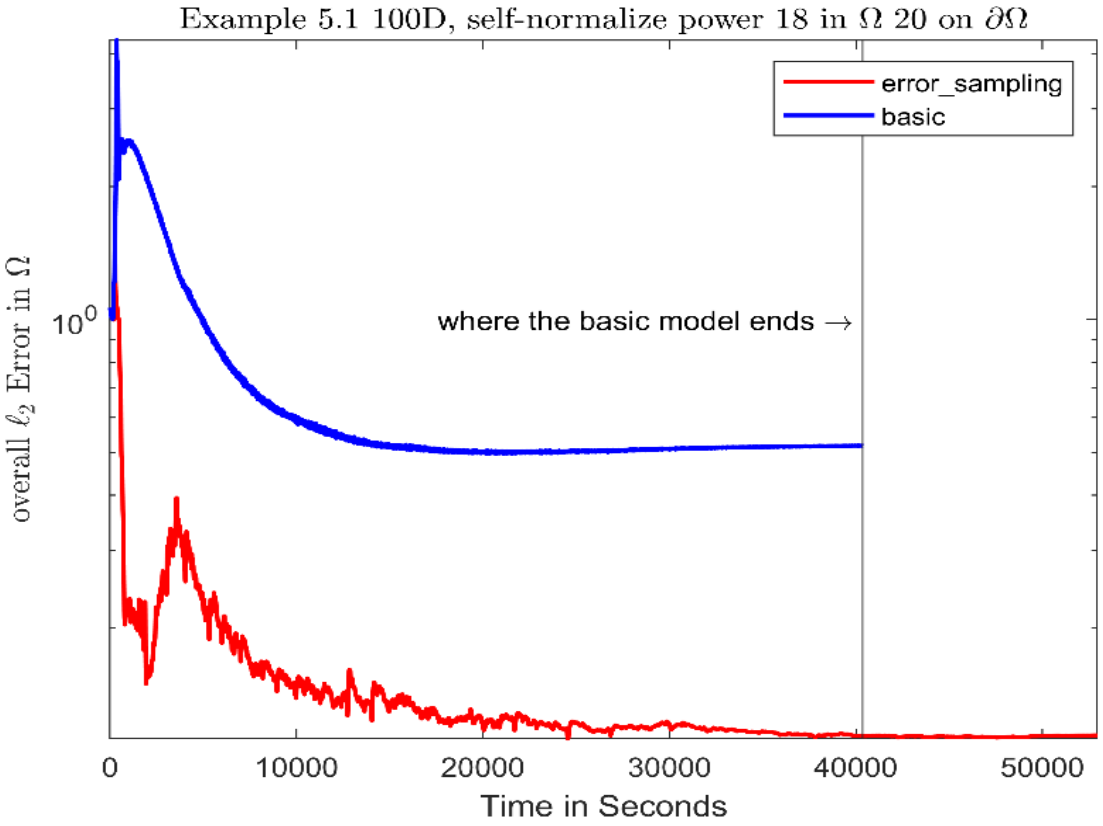


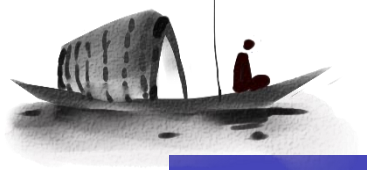
Error Decay Plot

max error v.s. epoch/iteration



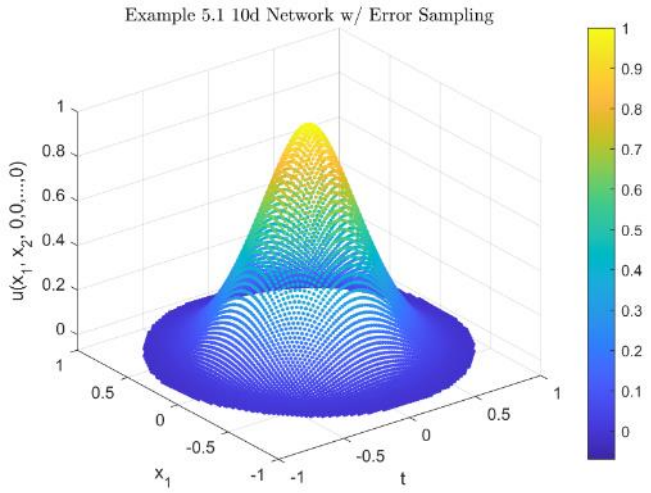
overall error v.s. training time



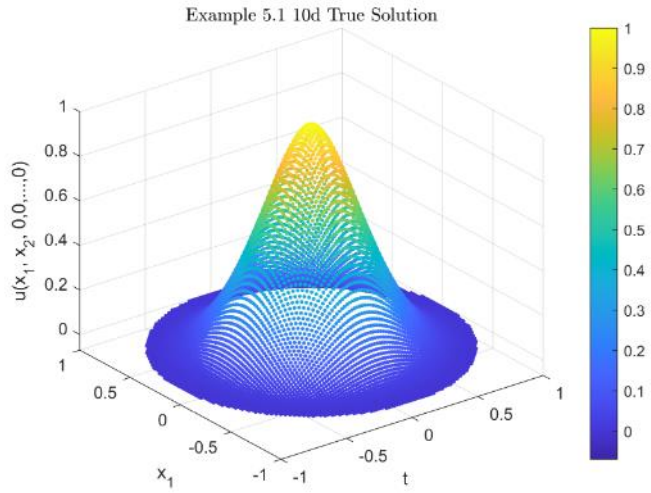


Visualization: $x_1 - x_2$ surface

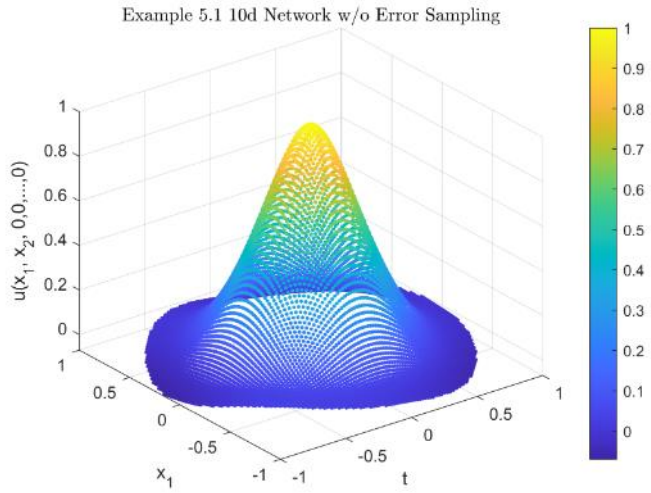
Figure 5.2: 5.1 10D $(x_1, x_2, 0, 0, ..., 0)$-surface of network solutions and the true solution



(a) With Error Sampling



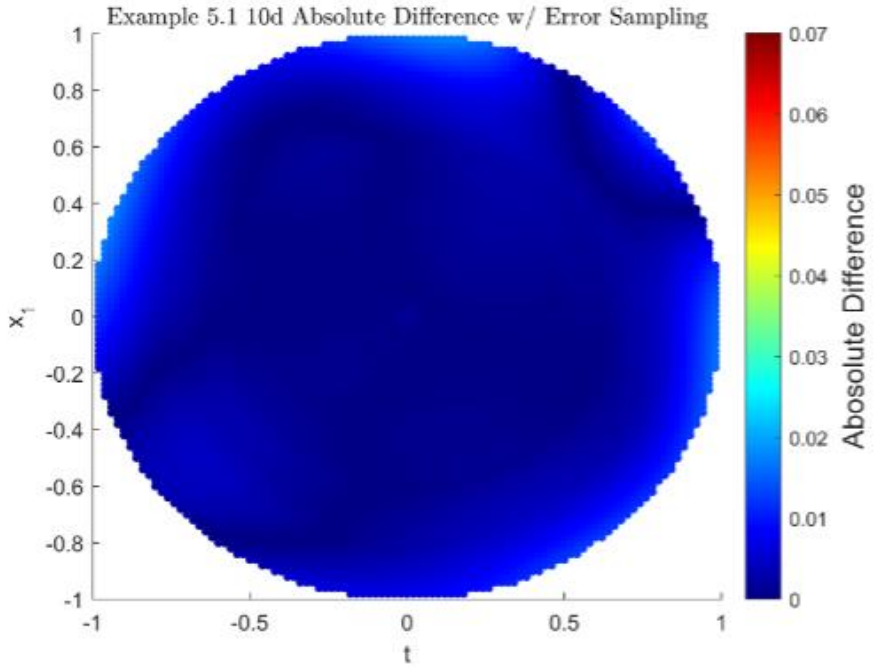
(b) The True Solution



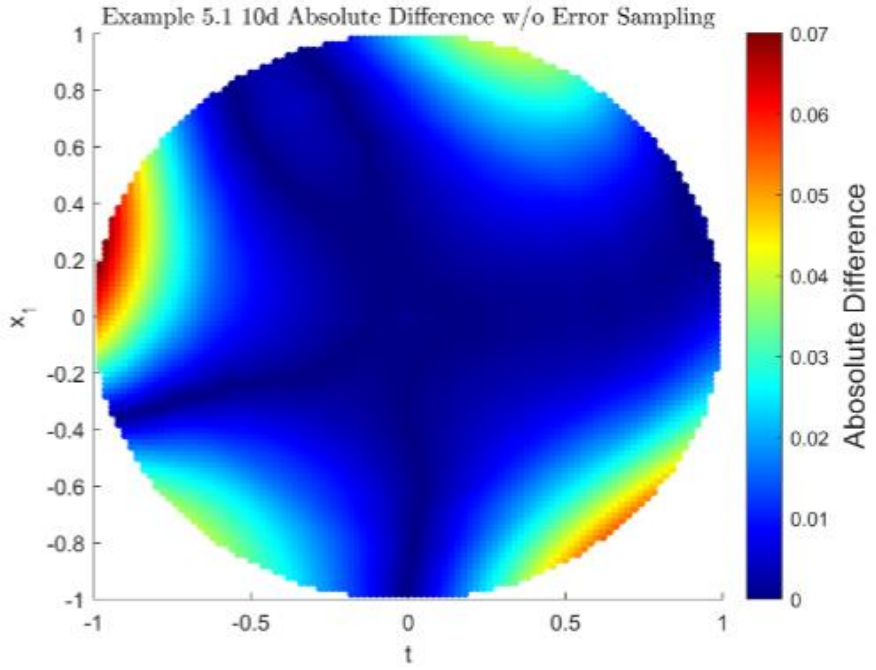
(c) W/O Error Sampling

Absolute Difference Btween Network Solutions and the True Solution

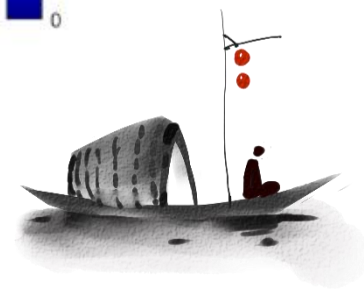
Figure 5.3: 5.1 10D $(x_1, x_2, 0, 0, ..., 0)$-surface Absolute Difference $|u - \phi|$



(a) With Error Sampling

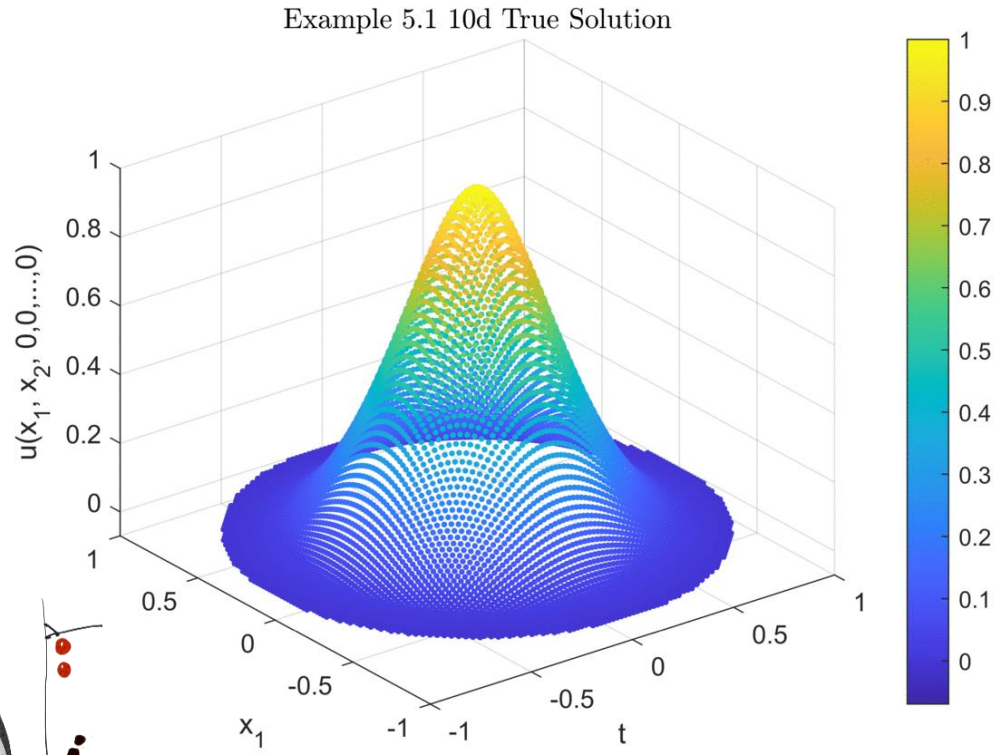


(b) Without Error Sampling

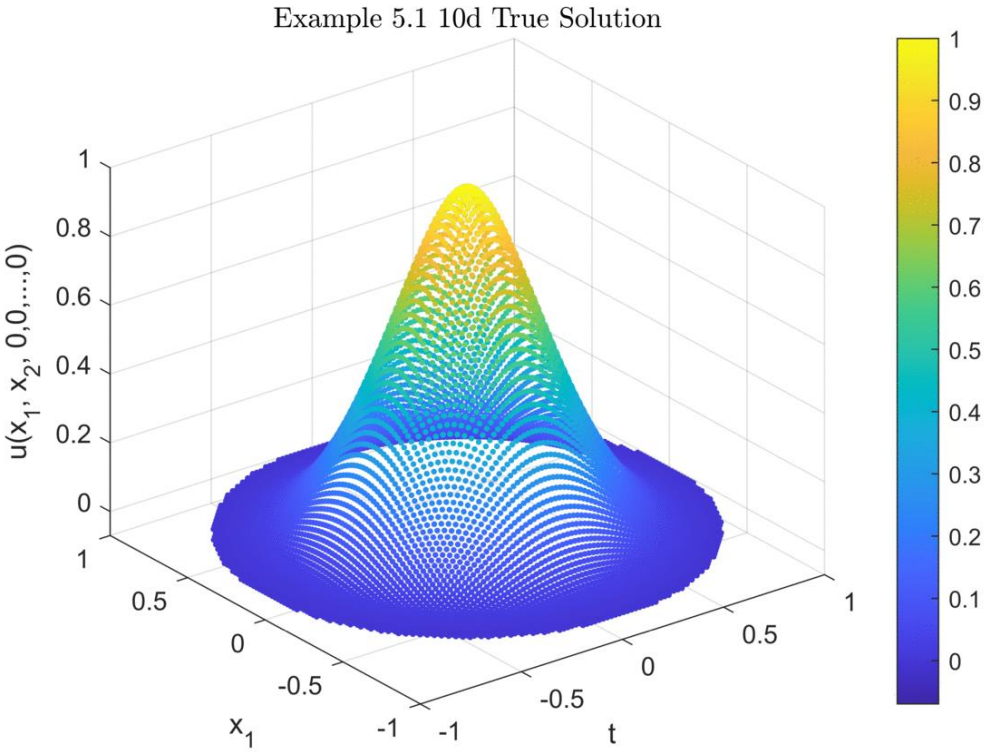


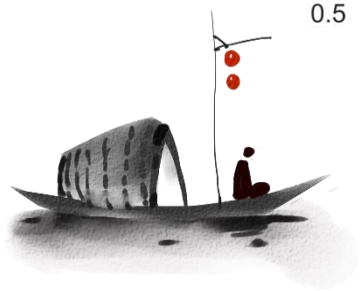
Animated Visualization

Exact solution v.s. error sampling



Exact solution v.s. without error sampling





Numerical Examples

- Next, parabolic equation(5.2):

$$\partial_t u(x,t) - \nabla_x \cdot \left((1 + \frac{1}{2}|x|) \nabla_x u(x,t) \right) = f(x,t), \quad \text{in } \Omega := \omega \times \mathbb{T}$$
$$u(x,t) = g(x,t), \quad \text{on } \partial\Omega = \partial\omega \times \mathbb{T}$$
$$u(x,0) = h(x), \quad \text{in } \omega,$$

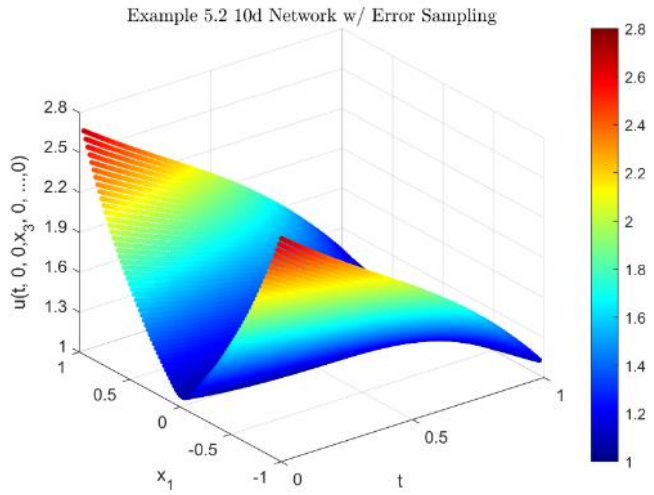
, where $\omega := \{x : |x| < 1\}$, $\mathbb{T} = (0,1)$, $g(x) = e^{|x|\sqrt{1-t}}$, and $h(x) = exp(|x|)$.

$f(x)$ is specified appropriately so that the exact solution is:

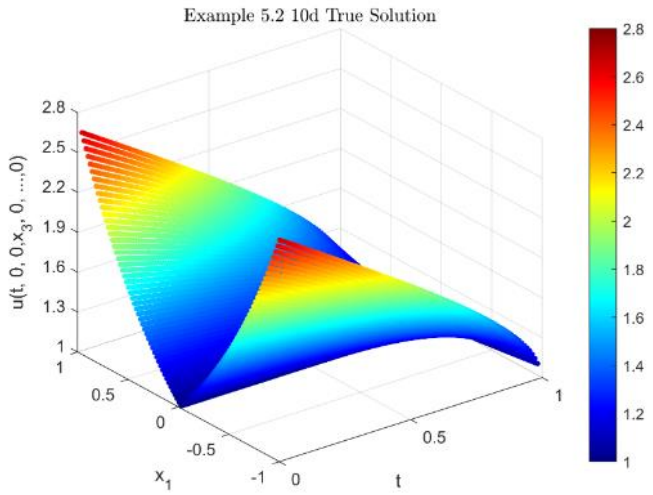
$$u(x,t) = e^{|x|\sqrt{1-t}}$$

Visulization: $t - x_3$ surface

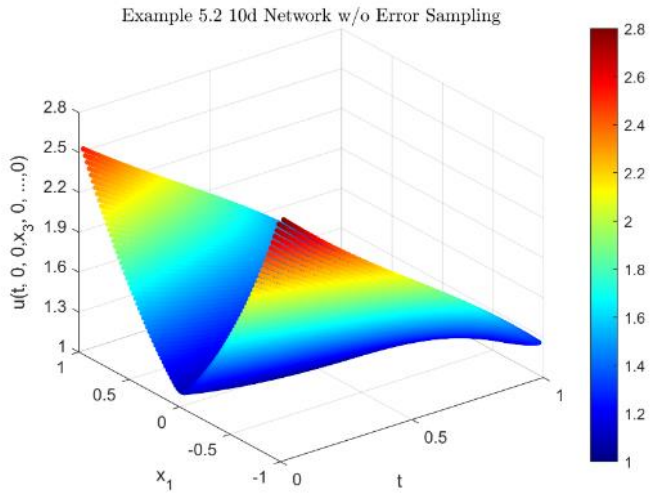
Figure 5.7: 5.2 10D $(t, 0, 0, x_3, 0, , ..., 0)$-surface of network solutions and the true solution



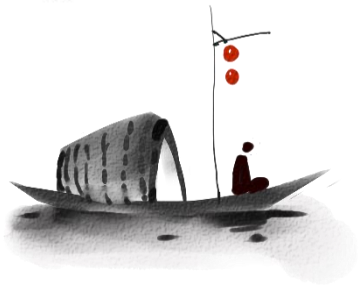
(a) With Error Sampling



(b) The True Solution

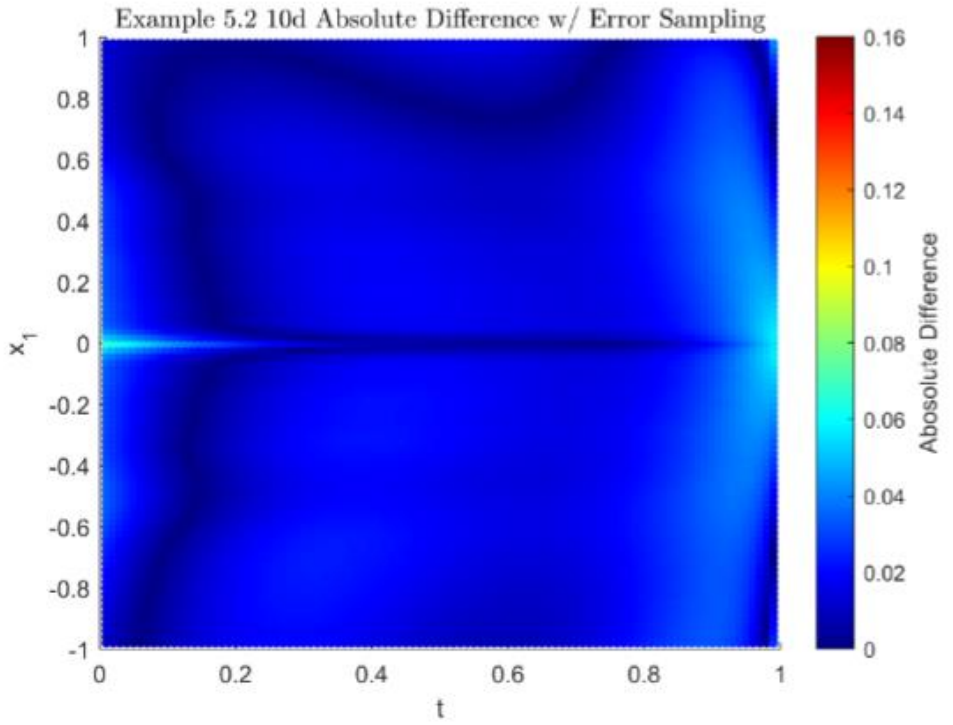


(c) W/O Error Sampling

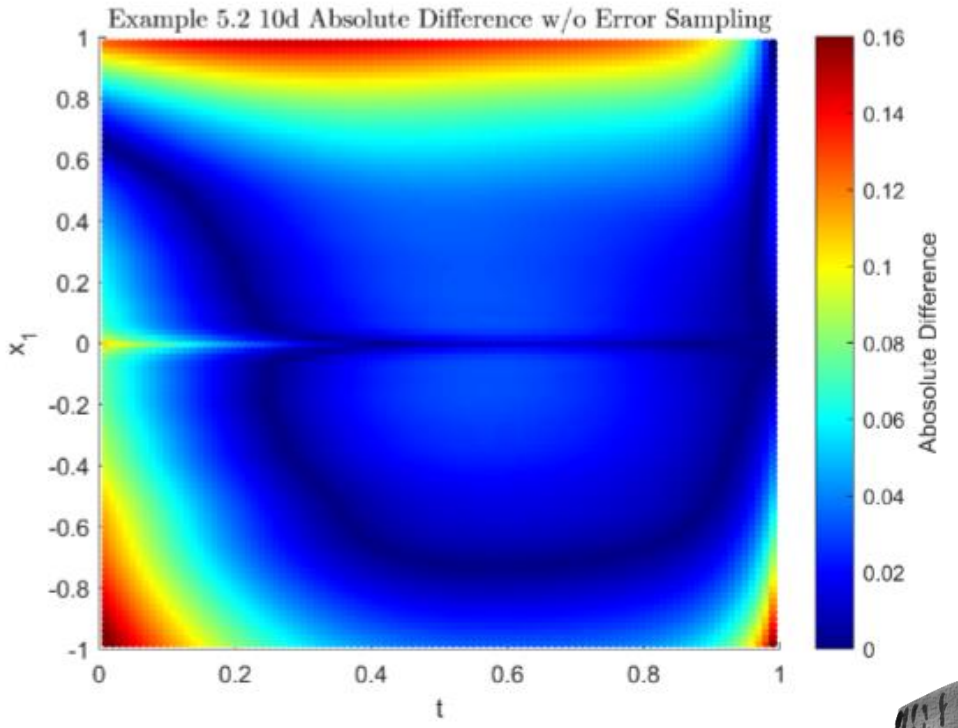


Absolute Difference Btween Network Solutions and the True Solution

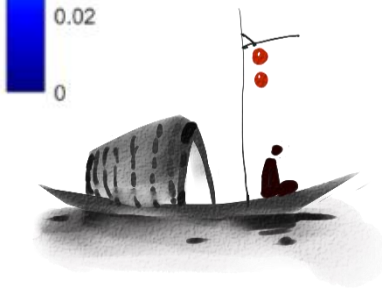
Figure 5.8: 5.2 10D $(t, 0, 0, x_3, 0, , ..., 0)$-surface Absolute Difference $|u - \phi|$



(a) With Error Sampling

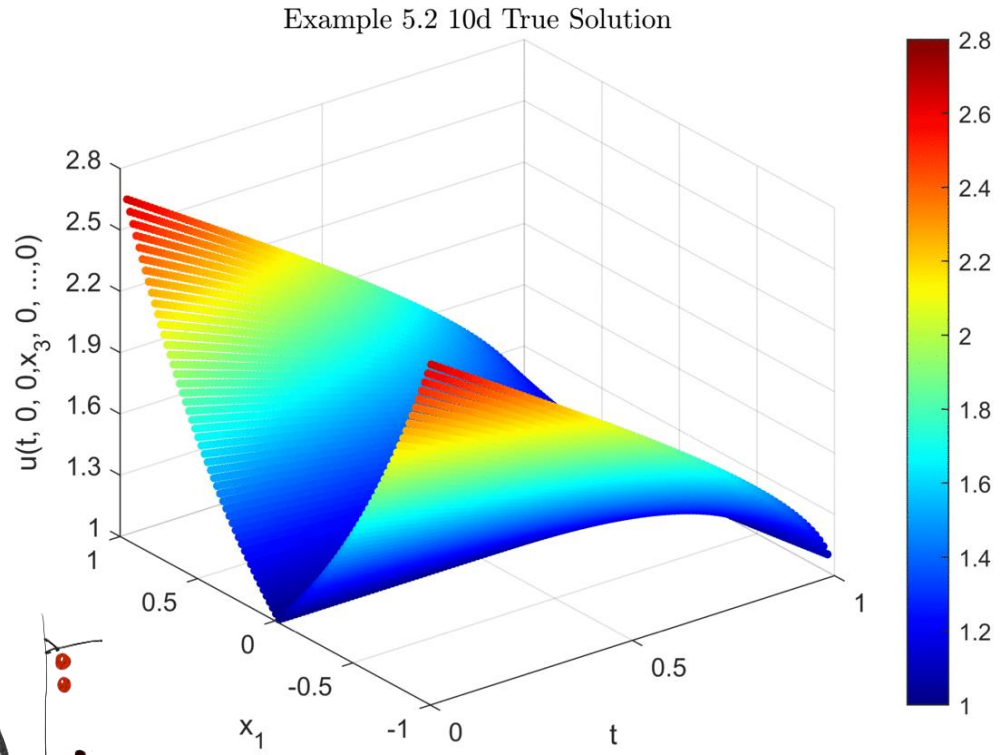


(b) Without Error Sampling

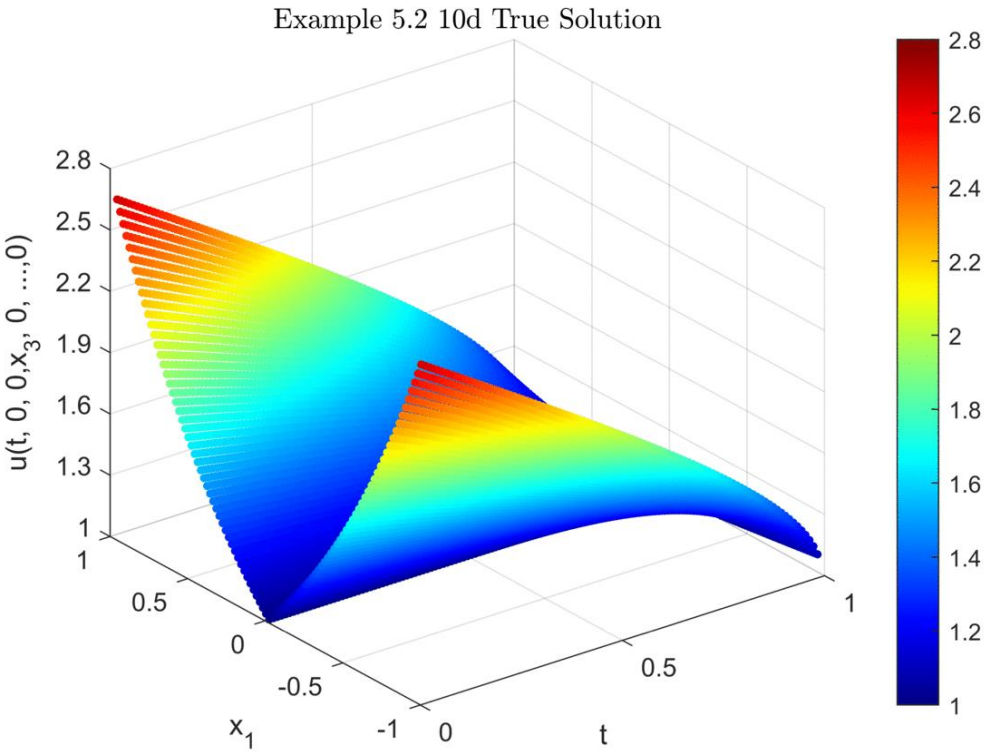


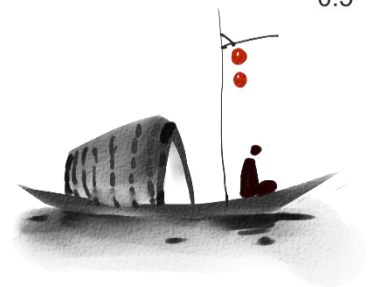
Animated Visualization

Exact solution v.s. error sampling



Exact solution v.s. without error sampling



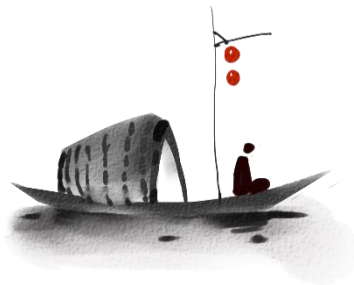


Compatibility Test

- Some recent frameworks, WAN[1], DGM[2], and DRM[3], are equipped with error sampling to test if they can achieve lower error within a certain time. Table below shows statistics of errors achieve in 30 different trials(with different random seeds).

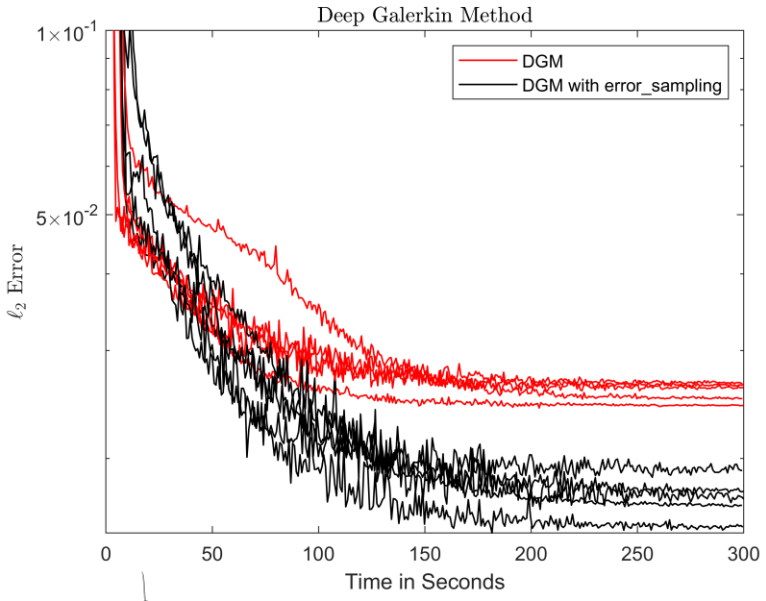
Framework	Mean	Standard Deviation	Minimum Value	Coefficient of Variation
DGM	0.0333	0.0064	0.0244	19.2%
DGM*	0.0280	0.0079	0.0155	29.2%
DRM	0.0273	0.0097	0.0132	35.5%
DRM*	0.0255	0.0093	0.0122	36.5%
WAN	0.0329	0.0062	0.0245	18.8%
WAN*	0.0282	0.0075	0.0153	26.6%

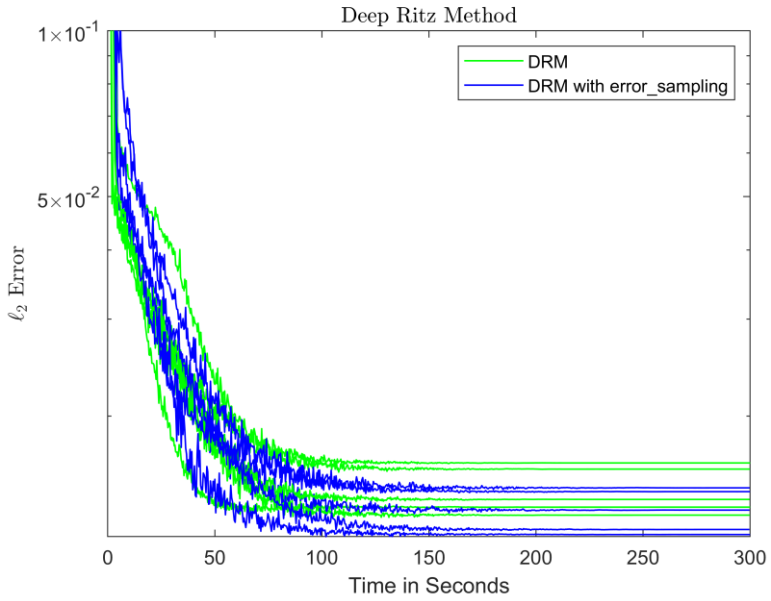
Table 6: Compatibility test

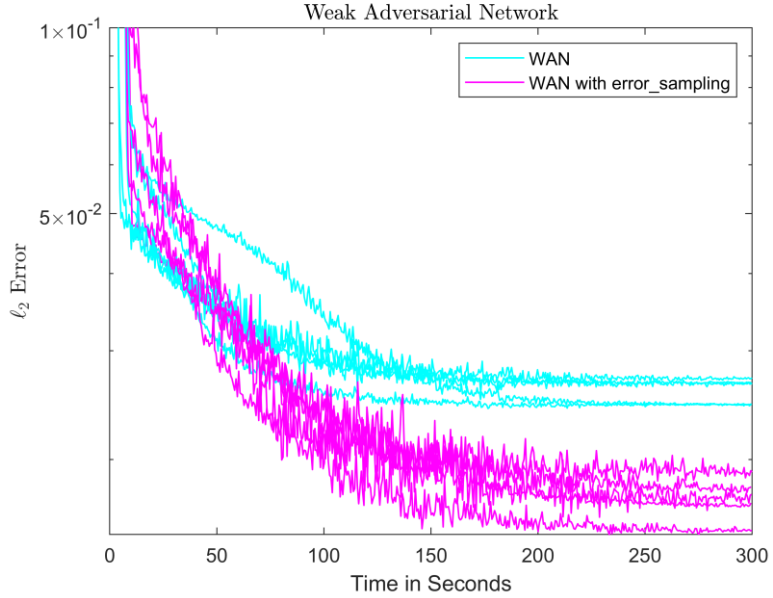


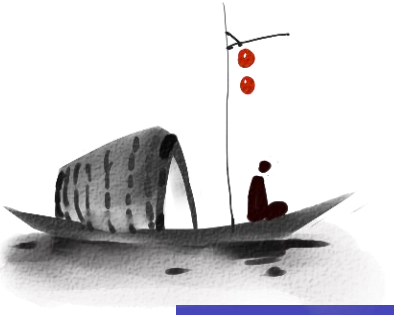
[1] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics
[2] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. Journal of computational physics, 375:1339–1364, 2018. ISSN 0021-9991.
[3] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 6(1), 2018. ISSN 2194-6701.

Plot: 5 Trials with Lowest Error in 30 Trials









Thank you

- This work is supported by the National Science Foundation under Grant No. DMS-2050133(REU Site: Math-ematical, Statistical, and Computational Methods in the Life Sciences).
- Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of theNational Science Foundation.