Adversarial Adaptive Sampling: A New Unified Adaptive Solver for PDEs

Jiayu Zhai

ShanghaiTech University

2025

PINN: Solving PDEs Using Neural Networks

We consider a general PDE problem on a compact domain Ω is to find $u \in \mathcal{F} : \Omega \mapsto \mathbb{R}$ in a proper function space $\mathcal{F}$, such that

$$\begin{aligned} \mathcal{L}u(\boldsymbol{x}) &= s(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega \\ \mathfrak{b}u(\boldsymbol{x}) &= g(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \partial\Omega, \end{aligned}$$

where $\mathcal{L}$ is a partial differential operator (e.g., the Laplace operator Δ), $\mathfrak{b}$ is a boundary operator (e.g., the Dirichlet boundary), s is the source function, and g represents the boundary conditions.

PINN: Solving PDEs Using Neural Networks

Among all parameterized neural networks $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ with the same architecture, to determine one that best fits the PDE, PINN forces the residuals of the PDE and its condition to be zero at the selected collocation points by minimizing the loss functional

$$\begin{aligned} J\left(u_{\boldsymbol{\theta}}\right) &= J_r(u_{\boldsymbol{\theta}}) + \gamma J_b(u_{\boldsymbol{\theta}}) \quad \text{with} \\ J_r(u_{\boldsymbol{\theta}}) &= \int_{\Omega} |r(\boldsymbol{x};\boldsymbol{\theta})|^2 d\boldsymbol{x} \\ J_b(u_{\boldsymbol{\theta}}) &= \int_{\partial\Omega} |b(\boldsymbol{x};\boldsymbol{\theta})|^2 d\boldsymbol{x}, \end{aligned}$$

where $r(\boldsymbol{x};\boldsymbol{\theta}) = \mathcal{L}u_{\boldsymbol{\theta}}(\boldsymbol{x}) - s(\boldsymbol{x})$, and $b(\boldsymbol{x};\boldsymbol{\theta}) = \mathfrak{b}u_{\boldsymbol{\theta}}(\boldsymbol{x}) - g(\boldsymbol{x})$, respectively, and $\gamma > 0$ is a penalty parameter.

PINN: Solving PDEs Using Neural Networks

In practice, the loss functional must be approximated numerically at the selected collocation points $\mathsf{S}_\Omega = \{\boldsymbol{x}_\Omega^{(i)}\}_{i=1}^{N_r}$ and $\mathsf{S}_{\partial\Omega} = \{\boldsymbol{x}_{\partial\Omega}^{(i)}\}_{i=1}^{N_b}$, that is, it is considered as a Monte Carlo approximation

$$J_N\left(u_{\boldsymbol{\theta}}\right) = \frac{1}{N_r}\sum_{i=1}^{N_r} r^2(\boldsymbol{x}_\Omega^{(i)};\boldsymbol{\theta}) + \gamma\frac{1}{N_b}\sum_{i=1}^{N_b} b^2(\boldsymbol{x}_{\partial\Omega}^{(i)};\boldsymbol{\theta}),$$

Strength of PINN

- One strength of PINN is of course providing an efficient solver for high dimensional PDEs.
- The other strenghth of PINN is that the data and PDE can collaborate to solve the problem. In the training process, the network learns the physics rule it should follow as the solution of the PDE; and its value, profile, and thus uniqueness, from data.

Strength of PINN: Example

Consider an SDE

$$\begin{cases} dx = (-4x(x^2+y^2-1)+y)dt + dW_t^x \\ dy = (-4y(x^2+y^2-1)-x)dt + dW_t^y \end{cases},$$

where dW_t^x and dW_t^y are independent Wiener processes. The stationary distribution u of the solution process satisfies the steady-state Fokker-Planck equation

$$\frac{1}{2}\Delta u - (f_x u)_x - (f_y u)_y = 0.$$

The exact solution is

$$u = Ke^{-2V/\epsilon^2} \quad \text{with } V = (x^2+y^2-1)^2.$$

Strength of PINN: Example

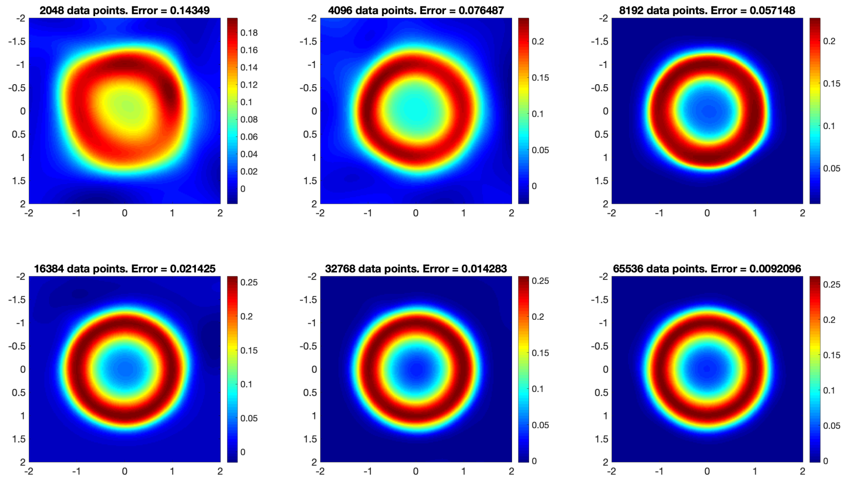


Figure 1: Solving an FPE with only data.

Strength of PINN: Example

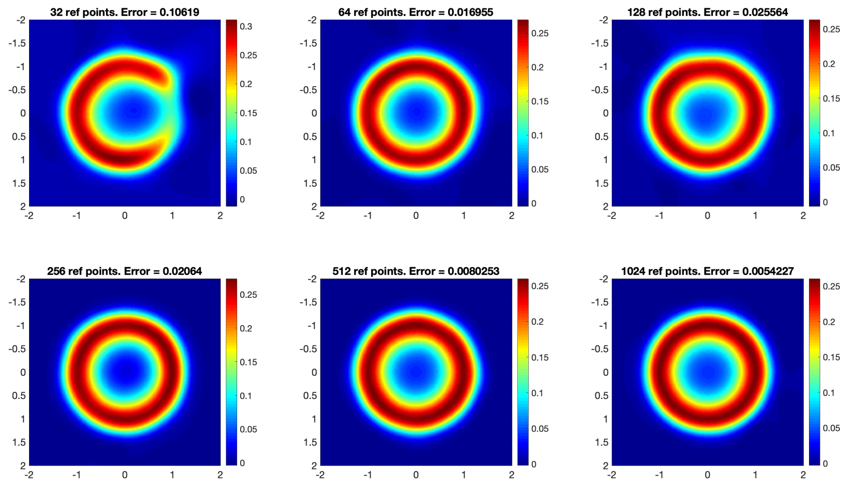


Figure 2: Solving an FPE with PINN and data.

Selecting Training Collocation Points: Motivation

As discussed above, we only need sparse data and collocation points for solving PDEs with PINN. So we need to choose them carefully.

- Most of collocation points must be selected in the regions where they are needed, namely, those are representative of the feature of the solution profile.
- Some collocation points must be selected from other regions, particularly stay in the historical needed regions, for the network to learning what's happening there.

If Fail to Select Representative Collocation Points

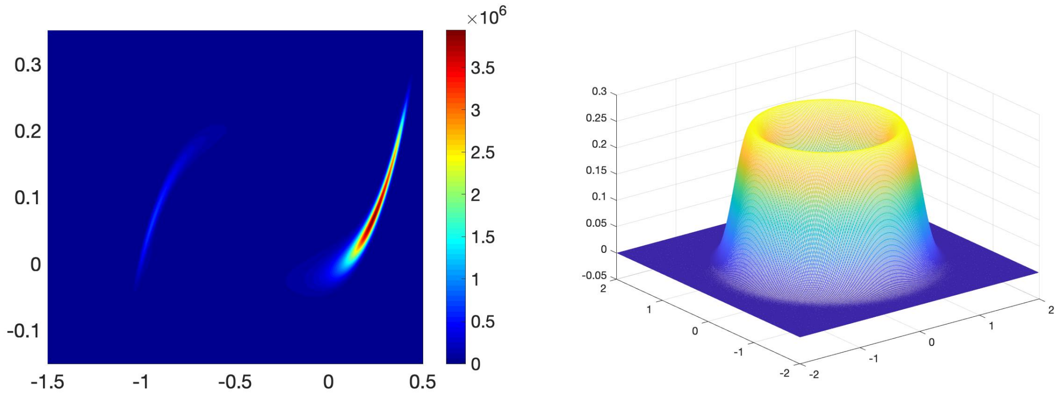


Figure 3: Steady state solutions of the Fokker-Planck equations (FPE) of an MMO system (left) and a potential-rotation system (right), respectively.

Viewpoint via Variance Reduction

In [Tang, Wan and Yang, 2023, JCP], adaptive sampling for deep learning methods are established and it is explained via importance sampling

$$J_r(u(x;\Theta)) = \int_\Omega \frac{r^2(x;\Theta)}{p(x)} p(x)\, dx = \frac{1}{N}\sum_{i=1}^{N} \frac{r^2(x_i;\Theta)}{p(x_i)}.$$

If we sample from some $p(x)$ instead of uniform distribution, the variance of $r^2(X)p^{-1}(X)$ can be reduced.
Then the accuracy of the Monte Carlo approximation will be improved for a fixed sample size, which yields a more accurate solution of PDEs, and so will the statistical error.

DAS: Adaptive Sampling by Posterior Residual Estimation

In [Tang, Wan and Yang, 2023, JCP], an adaptive sampling method, called deep adaptive sampling (DAS), based on posterior residual estimation was introduced. It is an analogue to adaptive or moving mesh finite element method, in which the mesh points are added or moved to the regions with larger residual.

DAS: Adaptive Sampling by Posterior Residual Estimation

- Every time an approximation is learnt with a certain distribution, a new residual distribution is generated.
- DAS learns this new residual distribution with a generative model. It builds a one-to-one mapping between this new residual distribution and an easy-to-sample distribution.
- Then it is used to sample collocation points for the next learning step.
- It is adaptive since this distribution represents where the statistical error is large or small, and thus where learning collocation points are needed.

Wasserstein GAN and Optimal Transport

The *Wasserstein distance* $d_{W^d}(\mu, \nu)$ between to probability measures μ and ν on a metric space X equipped with metric $d(\boldsymbol{x}, \boldsymbol{y})$

$$d_{W^d}(\mu, \nu) = \inf_{\pi \in \Pi(X \times X)} \int_{X \times X} d(\boldsymbol{x}, \boldsymbol{y})\, d\pi(\boldsymbol{x}, \boldsymbol{y}),$$

where $\Pi(X \times X)$ is the collection of all probability measure on $X \times X$ such that

$$\pi(A \times X) = \mu(A), \quad \pi(X \times B) = \nu(B)$$

for all measurable sets $A, B \subset X$.

Wasserstein GAN and Optimal Transport

The dual form of the Wasserstein distance is

$$d_{W^M}(\mu,\nu) = \sup\left\{ \int_X \phi(\boldsymbol{x})\, d(\mu-\nu)(\boldsymbol{x}) \middle| \, \|\phi\|_{\mathsf{Lip}} \leq 1 \right\}.$$

where $\|\cdot\|_{\mathsf{Lip}}$ denotes the Lipschitz norm, defined as

$$\|\phi\|_{\mathsf{Lip}} = \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{|\phi(\boldsymbol{x})-\phi(\boldsymbol{y})|}{d(\boldsymbol{x},\boldsymbol{y})}.$$

It measures the difficulty to reform from one probability measure μ to another one ν.

Wasserstein GAN and Optimal Transport

Wasserstein GAN forces the generator to generate samples from data distribution by minimizing the Wasserstein distance between the data distribution and generator distribution, and solves the following min-max problem.

$$\min_{\substack{G:\tilde{X}=G(Z)\\ Z\sim\mathbb{P}}} \max_{\|f\|_{\mathsf{Lip}}\leq 1} \mathbb{E}_{X\sim\mathbb{P}_g}[f(X)] - \mathbb{E}_{\tilde{X}\sim\mathbb{P}_d}[f(\tilde{X})],$$

Meanwhile, the trained maximizer f will play the role of the best discriminator (or critics) for the specific data distribution $\mathbb{P}_d$.

GANs and PINN

There are several works that uses GANs to learn the latent dynamics or PDE laws for data, e.g., [Yang, Zhang, Karniadakis, 2018], [Yang, Perdikaris, 2018], [Yang, Daskalakis, Karniadakis, 2021], [Li, Zhang, Wang, Guan, Tao, 2022], [Zang, Bao, Ye, Zhou, 2020], [Zeng, Bryngelson, Schaefer, 2022], etc.

Unifying PDE Solver and Adaptive Sampler

We now consider the PINN residual problem in a general setup

$$
\begin{aligned}
& \int_\Omega r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x} \\
= \quad & \int_\Omega r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x} - \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\int_\Omega p(\boldsymbol{x})d\boldsymbol{x} \\
& + \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\int_\Omega p(\boldsymbol{x})d\boldsymbol{x} \\
= & \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\left(\qquad \left[\int_\Omega p(\boldsymbol{x})d\mu_r - \int_\Omega p(\boldsymbol{x})d\mu_u\right] + \qquad \int_\Omega p(\boldsymbol{x})d\boldsymbol{x}\right)
\end{aligned}
$$

Unifying PDE Solver and Adaptive Sampler

Taking supremum on $V := \{p(\boldsymbol{x}) | \|p\|_{\mathsf{Lip}} \leq 1,\, 0 \leq p(\boldsymbol{x}) \leq M\}$,

$$
\begin{aligned}
&\sup_{p\in V} \int_\Omega r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x} \\
=&\sup_{p\in V} \int_\Omega r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x} - \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x} \int_\Omega p(\boldsymbol{x})d\boldsymbol{x} \\
&+ \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x} \int_\Omega p(\boldsymbol{x})d\boldsymbol{x} \\
\leq& \int_\Omega r^2(\boldsymbol{x};\theta)d\boldsymbol{x} \left(\sup_{p\in V} \left[\int_\Omega p(\boldsymbol{x})d\mu_r - \int_\Omega p(\boldsymbol{x})d\mu_u \right] + \sup_{p\in V} \int_\Omega p(\boldsymbol{x})d\boldsymbol{x} \right) \\
\leq& (d_{W^M}(\mu_r,\mu_u) + M) \int_\Omega r^2(\boldsymbol{x};\boldsymbol{\theta})\, d\boldsymbol{x},
\end{aligned}
$$

Unifying PDE Solver and Adaptive Sampler

This inequality

$$\sup_{p\in V}\int_\Omega r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x} \leq (d_{W^M}(\mu_r,\mu_u)+M)\int_\Omega r^2(\boldsymbol{x};\boldsymbol{\theta})\,d\boldsymbol{x}$$

forms our unified solver

$$\inf_u \sup_{p\in\hat{V}} \mathcal{J}(u,p) = \int_\Omega r^2(u(\boldsymbol{x}))p(\boldsymbol{x})\,d\boldsymbol{x},$$

Main Result

Assumption

The operator r is a surjection from a function space $E_1(\mathbb{R}^D)$ to $C_c^\infty(\Omega)$, the class of C^∞ functions that are compactly supported on Ω.

In general, $E_1(\mathbb{R}^D)$ can be any function space, such as space of neural networks, smooth functions, or Sobolev spaces. And this assumption means for any smooth function $f \in C_c^\infty(\Omega)$, equation $r^2(u^*) = f$ admits some solution u^*.

Main Result

Theorem

Let μ be the uniform probability distribution on Ω. Then the optimal value of the min-max problem is 0. Moreover, there is a sequence $\{u_n\}_{n=1}^{\infty}$ of functions with $r(u_n) \neq 0$ for all n, such that

$$\lim_{n\to\infty} \mathcal{J}(u_n) = \lim_{n\to\infty} \int_{\Omega} r^2(u_n(\boldsymbol{x}))p(\boldsymbol{x})d\boldsymbol{x} = 0.$$

Meanwhile, it has the following properties:

1. *The residual sequence $\{r(u_n)\}_{n=1}^{\infty}$ converges to 0 in $L^2(d\mu)$.*
2. *The renormalized squared residual distributions*

$$d\nu_n \triangleq \frac{r^2(u_n)}{\int_{\Omega} r^2(u_n(\boldsymbol{x}))\, d\boldsymbol{x}}\, d\mu$$

converge to the uniform distribution μ in d_{W^M}.

Replacement of the Boundedness Condition

We proved in this work that the "test" function space

$$V := \{p(\boldsymbol{x}) | \|p\|_{\text{Lip}} \leq 1,\ 0 \leq p(\boldsymbol{x}) \leq M\}$$

can be replaced with

$$\hat{V} = \{p(\boldsymbol{x}) | \|p\|_{\text{Lip}} \leq 1,\ p(\boldsymbol{x}) \geq 0,\ \int_{\Omega} p(\boldsymbol{x}) d\boldsymbol{x} = 1\},$$

which means the "test" functions are really probability distributions as proposed at the beginning, namely, they perform as **adaptive samplers**.

Discussion on the Regularity Constraint

The regularity constraint $\|p\|_{\text{Lip}} \leq 1$ is important. Without this constraint, maximizing $\int_\Omega r^2(u(\boldsymbol{x}))p(\boldsymbol{x})d\boldsymbol{x}$ over p will give

$$p = \delta_{\boldsymbol{x}_0}, \quad \text{where } \boldsymbol{x}_0 = \arg\max_{\boldsymbol{x}} r^2(u(\boldsymbol{x})).$$

So p can

- ONLY figure out where the residual has the largest value and the samples are most needed,
- but NOT figure out the residual profile, that is, how the residual can be efficiently transported to an "evenly" spread zero.

Implementation of the Regularity Constraint of AAS

- For this regularity condition $\|p\|_{\text{Lip}} \leq 1$, we didn't use the strategy in
  - WGAN that bounds the parameters in u, or
  - the improved WGAN that forces the gradient on the "diagonal" to be 1;
- For differentiable functions, the condition is equivalent to $\|\nabla_{\boldsymbol{x}} p\|_\infty \leq 1$.
- We use a regularization term $-\beta \int_\Omega |\nabla_{\boldsymbol{x}} p_{\boldsymbol{\alpha}}(\boldsymbol{x})|^2 d\boldsymbol{x}$ in the min-max problem, to make the gradient as small as possible.
- So we actually use a weaker condition.

Further Discussion on the Regularity Constraint

Consider the following variation problem:

$$\min_{p_{\boldsymbol{\alpha}}>0} \mathcal{L}(p_{\boldsymbol{\alpha}}) = \beta \int_{\Omega} |\nabla_{\boldsymbol{x}} p_{\boldsymbol{\alpha}}|^2 d\boldsymbol{x} - \int_{\Omega} r^2(\boldsymbol{x};\boldsymbol{\theta}) p_{\boldsymbol{\alpha}}(\boldsymbol{x}) dx + \lambda \left(\int_{\Omega} p_{\boldsymbol{\alpha}}(d\boldsymbol{x}) - 1 \right).$$

It leads to the Lagrange equation

$$\begin{cases} 2\beta \nabla^2 p_{\boldsymbol{\alpha}}(\boldsymbol{x}) + r^2(\boldsymbol{x};\boldsymbol{\theta}) - \lambda = 0, & \boldsymbol{x} \in \Omega, \\ \frac{\partial p_{\boldsymbol{\alpha}}}{\partial \boldsymbol{n}} = 0, & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{1}$$

with

$$\lambda = \frac{1}{|\Omega|} \int_{\Omega} r^2(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x}.$$

Further Discussion on the Regularity Constraint

With enough regularity assumptions, its solution satisfies

$$\|p_{\boldsymbol{\alpha}}\|_{H^{k+2}(\Omega)} \leq C\|f\|_{H^k(\Omega)},$$

where $f(\boldsymbol{x}) = (\lambda - r^2)/(2\beta)$.
According to the Sobolev Imbedding Theorem,

$$W^{k,1}(\Omega) \to C^{0,1}(\overline{\Omega}),$$

when $D = k - 1$. Thus up to a set of measure zero, we have

$$\|p_{\boldsymbol{\alpha}}\|_{C^{0,1}(\overline{\Omega})} \leq C_1\|p_{\boldsymbol{\alpha}}\|_{W^{k,1}(\Omega)} \leq C_2\|p_{\boldsymbol{\alpha}}\|_{H^k(\Omega)}.$$

So $p_{\boldsymbol{\alpha}}$ is Lipschitz continuous.

Discussion on the Integral Constraint: the Role of p

The other two constraints

$$p(\boldsymbol{x}) \geq 0 \quad \text{and} \quad \int_\Omega p(\boldsymbol{x}) d\boldsymbol{x} = 1$$

in the space $\hat{V}$ exactly mean p is a probability density function.

This is what we really need for adding p here as the generator for adaptive samples!

Implementation of p in AAS

- For the adaptive sampler p, we use a flow-based model called KRnet, which builds a bijection between an objective distribution and an easy-to-sample, such as normal or uniform, distribution p_e.
- In the training process, it is trained simultaneously and maps p_e samples to residual samples.

Further Discussion on the Role of p

In AAS, p plays two different roles:

- In the proof of the theorem, p plays the role of **discriminator** as in the Wasserstein distance or WGAN. It figures out the difference between two distributions, and then works as the best way to transport from one to the other according to this difference profile. This is the **similar** side with WGAN.
- In the implementation of the algorithm, p plays the role of **generator**. Since $p \in \hat{V}$, it is a probability distribution. And in fact it shows how the normalized residual distribution looks like: where it is large or small, where the adaptive samples are needed. This is the **different** side from WGAN.

Further Discussion on the Role of p

To sum up,

- as a discriminator, p will be large when the residual (so its difference with zero distribution) is large;
- as a generator, it generates more samples in the regions of large residuals.

Now, **adaptivity is achieved!**

Sampling Strategy

In case the training step of p loses information in low residual regions, we keep some samples from the last min-max step and update most with newly trained p, namely apply a mixture model

$$p(\boldsymbol{x}) = (1-\zeta)\tilde{p}(\boldsymbol{x}) + \zeta p_{\boldsymbol{\alpha}}(\boldsymbol{x}),$$

Further Applications for AAS

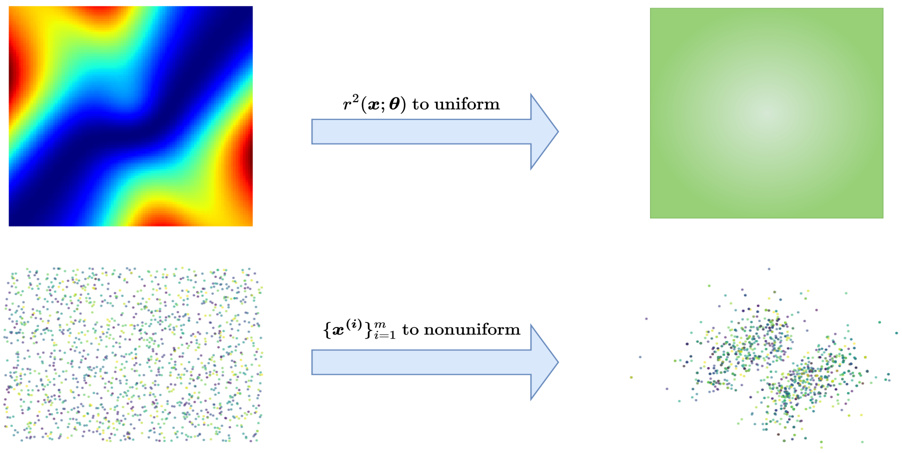
- One may noticed that AAS can be applied to other methods, not just for PINNs, for example, deep Ritz can also utilize it.
- Furthermore, it is not restricted to solving PDEs, but can be used to solve any problems in which adaptive sampling is needed.

Compare to DAS

- Now we don't need to separately train the generator using samples as in DAS.
- We just need to solve the maximization problem. It combines the two procedures together.
- The two model networks evolve simultaneously.

Performance of AAS

Two neural network models are simultaneously trained in the adversarial adaptive sampling framework.


$r^2(\boldsymbol{x};\boldsymbol{\theta})$ to uniform
$\{\boldsymbol{x}^{(i)}\}_{i=1}^m$ to nonuniform

The residual is minimized and finally becomes “uniform”, while the collocation points are updated and finally become nonuniform.

Numerical Examples I: Single Peak

The following is a benchmark test problem for adaptive finite element methods:

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= s(\boldsymbol{x}) \quad \text{in } \Omega, \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}) \quad \text{on } \partial\Omega, \end{aligned} \tag{2}$$

on $\Omega = [-1, 1]^2$ with the exact solution

$$u(x_1, x_2) = \exp\left(-1000[(x_1 - 0.5)^2 + (x_2 - 0.5)^2]\right),$$

which has a peak at $(0.5, 0.5)$ and decreases rapidly away from it.

Numerical Examples I: Single Peak

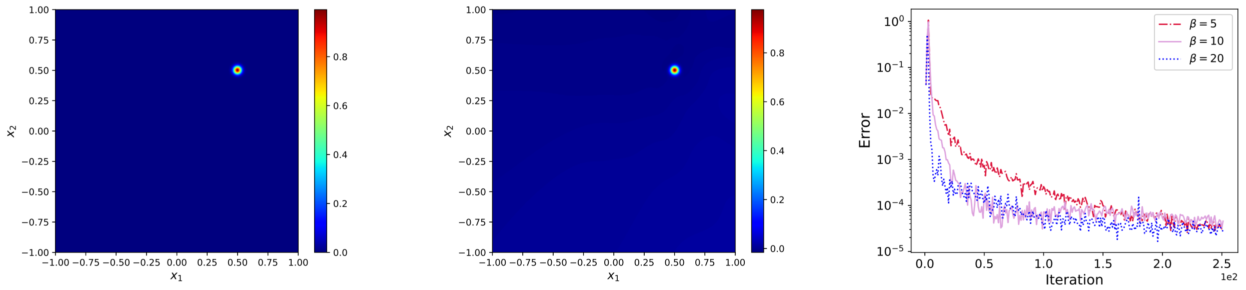
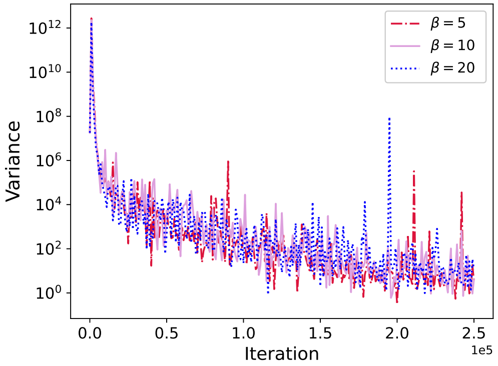


Figure 4: The results for the peak test problem. (a) The exact solution. (b) AAS approximation. (c) The error behaviour.

Numerical Examples I: Single Peak



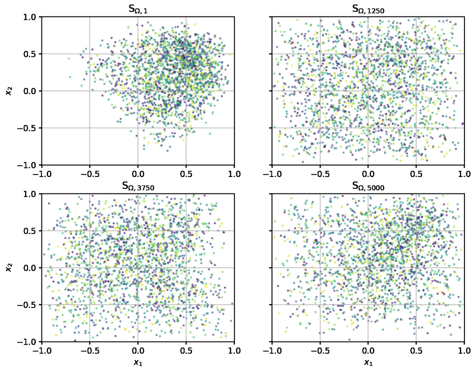


Figure 5: The evolution of the residual variance and the training set for the peak test problem. Left: The variance behavior. Right: The evolution of the training set.

Numerical Examples II: Double Peak

We generate the above example to one with a two-peak solution

$$\begin{aligned} -\nabla \cdot [u(\boldsymbol{x})\nabla v(\boldsymbol{x})] + \nabla^2 u(\boldsymbol{x}) = s(\boldsymbol{x}) \quad \text{in } \Omega, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{on } \partial\Omega, \end{aligned} \tag{3}$$

where $v(\boldsymbol{x}) = x_1^2 + x_2^2$ on $\Omega = [-1, 1]^2$ with the exact solution

$$u(x_1, x_2) = \mathrm{e}^{-1000[(x_1-0.5)^2+(x_2-0.5)^2]} + \mathrm{e}^{-1000[(x_1+0.5)^2+(x_2+0.5)^2]},$$

where the two peaks are at $(0.5, 0.5)$ and $(-0.5, -0.5)$.

Numerical Examples II: Double Peak

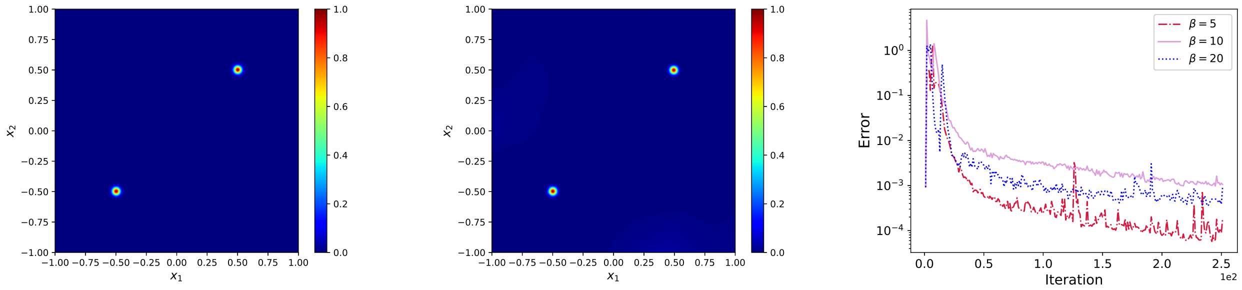
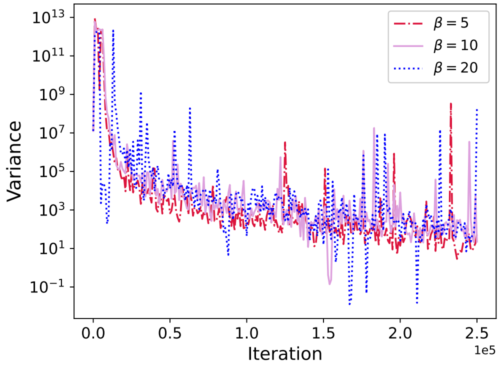


Figure 6: The results for the two-peak test problem. (a) The exact solution. (b) AAS approximation. (c) The error behavior.

Numerical Examples II: Double Peak



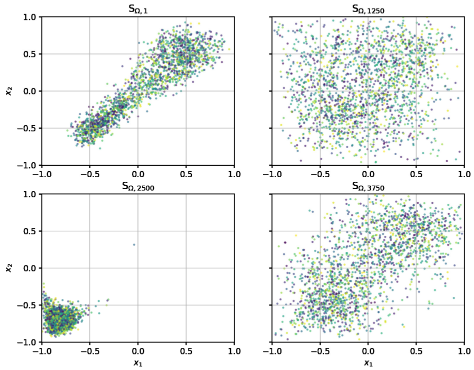


Figure 7: The evolution of the residual variance and the training set for the two-peak test problem. Left: The variance behavior. Right: The evolution of the training set.

Numerical Examples III: Parametric Burgers' Equation

We also test AAS on parametric PDEs that are commonly used in the design of engineering systems and uncertainty quantification.

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \nu[(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2] \quad x, y \in [0,1], \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \nu[(\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2] \quad t \in [0,1] \end{aligned}$$

where u and v are the velocities along x and y directions respectively, and $\nu \in (0,1]$ is a parameter that represents the kinematic viscosity of fluid. The exact solution is

$$\begin{aligned} u(x,y,t) &= \frac{3}{4} - \frac{1}{4[1+\exp((-4x+4y-t)/(32\nu))]}, \\ v(x,y,t) &= \frac{3}{4} + \frac{1}{4[1+\exp((-4x+4y-t)/(32\nu))]}, \end{aligned}$$

Numerical Examples III: Parametric Burgers' Equation

The problem setup space is $\boldsymbol{x} = [t, x, y, \nu]$, i.e., $D = 4$. When ν is small, solving this problem is quite challenging. We use AAS to train a neural network $u_{\boldsymbol{\theta}}(\boldsymbol{x})$ to approximate the solution over the entire space $\boldsymbol{x} = [t, x, y, \nu] \in [0, 1]^4$.

Numerical Examples III: Parametric Burgers' Equation

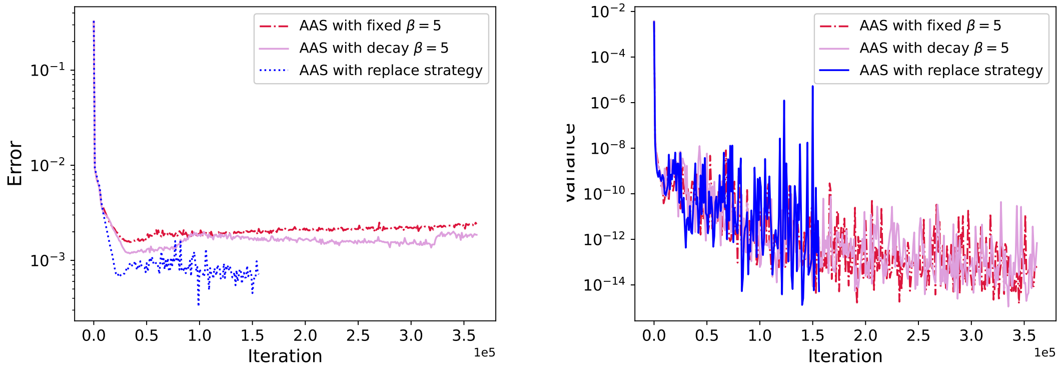


Figure 8: The results of the parametric Burgers' equation. Left: The error behavior. Right: The evolution of the variance.

Numerical Examples IV: High-dimensional nonlinear equation

In this part, we consider the following ten-dimensional nonlinear partial differential equation

$$\begin{aligned} -\Delta u(\boldsymbol{x}) + u(\boldsymbol{x}) - u^3(\boldsymbol{x}) &= s(\boldsymbol{x}), \quad \boldsymbol{x} \text{ in } \Omega = [-1,1]^{10} \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}), \quad \boldsymbol{x} \text{ on } \partial\Omega, \end{aligned} \tag{4}$$

with the exact solution $u(\boldsymbol{x}) = \mathrm{e}^{-10\|\boldsymbol{x}\|_2^2}$.

Numerical Examples IV: High-dimensional nonlinear equation

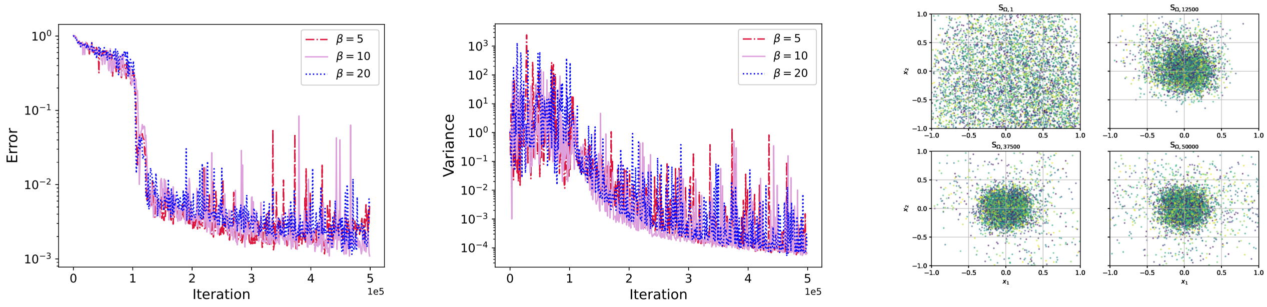


Figure 9: The results of the ten-dimensional nonlinear test problem. (a) The error behavior. (b) The variance behaviour. (c) The evolution of the training set, $x_1 - x_2$ plane ($\beta = 10$).

Numerical Examples IV: High-dimensional nonlinear equation

Table 1: Error comparison of adaptive sampling methods

Test problem	PINN	DAS-G	DAS-R	RAR	AAS
PDE (2)	9.74e-04	3.75e-04	1.93e-04	-	**2.97e-05**
PDE (3)	3.22e-02	1.51e-03	6.21e-03	-	**1.09e-04**
PDE (4)	1.01	9.55e-03	1.26e-02	9.83e-01	**1.31e-03**