ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

DMIS: Dynamic Mesh-based Importance Sampling for Training Physics-Informed Neural Networks

AAAI2023

Background

Physics systems are usually described by partial differential equations (PDEs).

In most cases, analytical solutions are not available.

- numerical algorithms
  - computationally expensive
  - face severe challenges in multiphysics and multiscale systems
  - data assimilation introduces additional uncertainties
- machine learning
  - Physics-informed Neural Networks (PINNs) encode PDEs into the loss function of neural networks

$$\frac{\partial u}{\partial t} + \mathcal{N}_{\boldsymbol{x}}[u] = 0, \boldsymbol{x} \in \Omega, t \in [0, T],$$
$$u(t, \boldsymbol{x})|_{t=0} = u_0(\boldsymbol{x}), \boldsymbol{x} \in \Omega,$$
$$u(t, \boldsymbol{x}) = g(t, \boldsymbol{x}), x \in \partial\Omega, t \in [0, T],$$

⇩ use $u_\theta(t, x)$ to approximate $u(t, x)$

$$\boldsymbol{\theta}^* \approx \arg\min_{\boldsymbol{\theta}} \mathcal{L}_f + \lambda_1 \mathcal{L}_i + \lambda_2 \mathcal{L}_b,$$

improvement directions:

- weights of loss terms
- parallel computation(domain decomposition)
- sampling

Importance Sampling

we want to calculate the expectation of $f(x)$, $x \sim p(x)$, we can get $E[f(x)]$ as follows:

$$E[f(x)] = \int f(x)p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

If we can't sample from $p(x)$, or if it's expensive to sample from $p(x)$, how do we get $E[f(x)]$

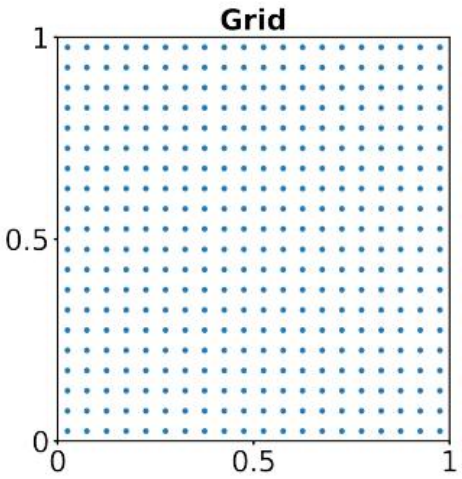
$$E[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)\frac{p(x_i)}{q(x_i)}$$

we define importance weight:

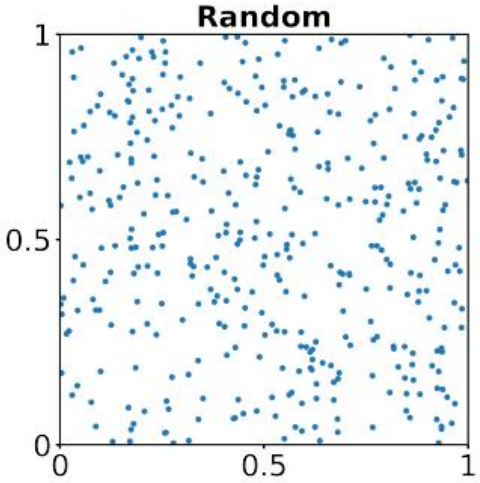
$$w(x) = \frac{p(x)}{q(x)}$$

Sampling Strategy

Equispaced uniform grid

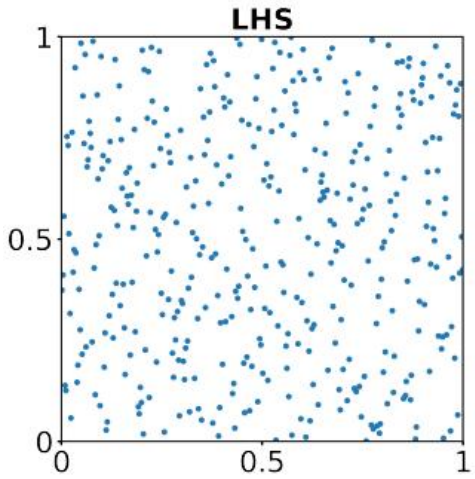


Random sampling



Latin hypercube sampling (LHS)

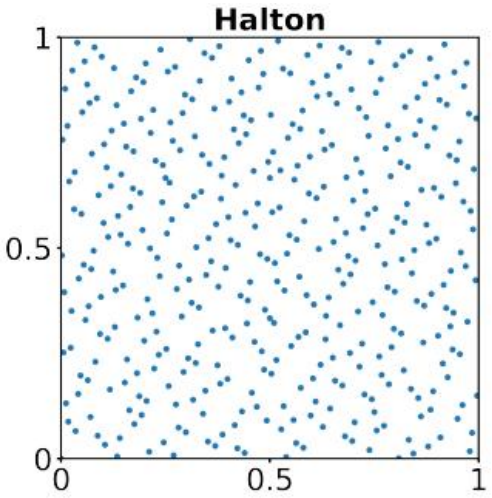
layering, sampling, disordering



Sampling Strategy

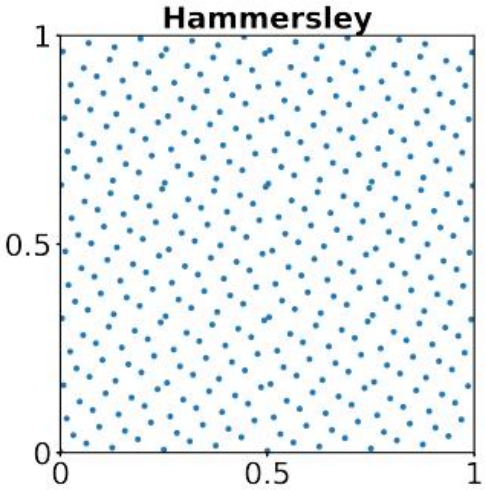
Halton sequence (Halton)

constructed according to a deterministic method that uses a prime number as its bases: $[0, 1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, \ldots]$



Hammersley sequence (Hammersley)

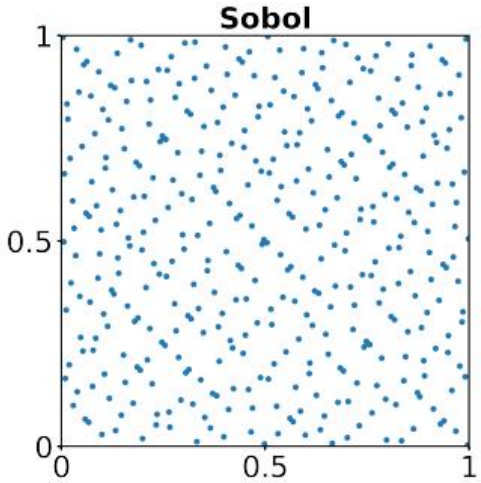
the same as the Halton sequence, except in the first dimension where points are located equidistant from each other

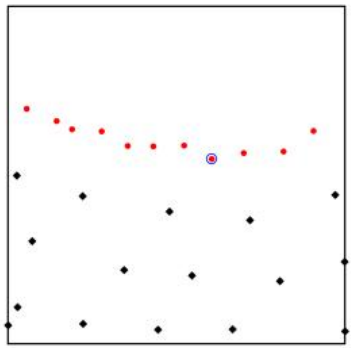


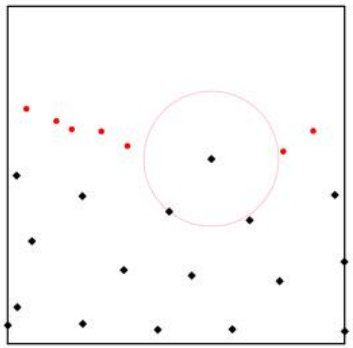
Sampling Strategy

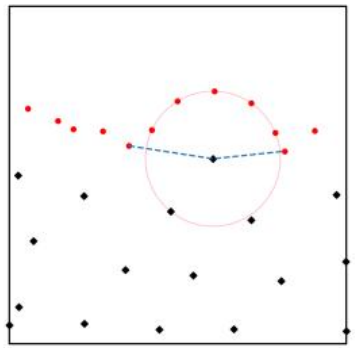
Sobol sequence (Sobol)

Let the nth sampling point be x_n, the binary representation of n is $\ldots b_3 b_2 b_1$, $x_n = b_1 v_1 \oplus b_2 v_2 \oplus \ldots$, where v_i is direction number, $v_i = m_i / 2^i$.









FF sampling

1. find the lowest node
2. Remove all the potential points with a distance less than r
3. two neighboring points determine an arc
4. Generate K new nodes across the arc

Method

Datasets for PDE residuals, initial conditions and boundary conditions are denoted by N_f, N_i and N_b, we only introduce importance sampling into the sampling of N_f

$$\mathcal{L}_f = \frac{1}{|N_f|}\sum_{i=1}^{|N_f|}\alpha_i \ell_f(\mathrm{x}_i;\boldsymbol{\theta}),\ \alpha_i = \frac{p_i}{q_i},$$

Considering points are obtained by uniform sampling in general

$$\alpha_i = \frac{1}{|N_f|q_i},\ i \in \{1,2,\cdots,|N_f|\},$$

the best sampling probability: $q^* \propto \|\nabla_{\boldsymbol{\theta}}\ell_f(\mathrm{x},\boldsymbol{\theta})\|_2$

$$q_i^{(t)} = \frac{\ell_f(\mathrm{x}_i,\boldsymbol{\theta}^{(t)})}{\sum_{j=1}^{N_f}\ell_f(\mathrm{x}_j,\boldsymbol{\theta}^{(t)})},\ i \in \{1,2,\cdots,|N_f|\},$$

data points with high loss lead to sharp local gradients

$$\alpha_i' = (\frac{1}{|N_f|q_i})^{\beta},\ \beta \in [1,+\infty),\ i \in \{1,2,\cdots,|N_f|\}.$$

To further reduce the computational cost, we dynamically generate a subset S from N_f.
calculate the sample weights of points in S exactly, and the weights of other points are obtained by interpolation.

$$g_i^{(t)} \propto \|q_i^{(t)} - q_i^{(t-1)}\|,\ i \in \{1, 2, \cdots, |N_f|\},$$

The interpolation based on Delaunay is time-consuming and it is also unnecessary to update the triangular mesh in each iteration step.

$$\mathrm{Sim}(\boldsymbol{v}^{(t_0)}, \boldsymbol{v}^{(t)}) = \frac{\boldsymbol{v}^{(t_0)} \cdot \boldsymbol{v}^{(t)}}{||\boldsymbol{v}^{(t_0)}|| \cdot \|\boldsymbol{v}^{(t)}\|},$$

Method

Algorithm 1: Sampler with DMIS

Input: batch size of PDE residuals $|M_f|$.
Parameter: set size of mesh points $|S|$, reweighting parameter β, mesh update threshold γ, dataset of PDE residuals N_f, iteration step t.
Output: mini-batch M_f, vector of sample weights $\boldsymbol{\alpha}'$.

Initialization

$t_0 \leftarrow 0$
$q_i^{(t_0)} \leftarrow 1/|N_f|, i \in \{1, 2, \cdots, |N_f|\}$
$g_i^{(t_0)} \leftarrow 1/|N_f|, i \in \{1, 2, \cdots, |N_f|\}$
$S \leftarrow |S|$ points sampled with $g_i^{(t_0)}$ from N_f
Build triangular mesh by Delaunay Triangulation

Mini-batch sampling

Compute $\{\ell_f(\mathrm{x}_i, \boldsymbol{\theta}^{(t-1)})\}_{\mathrm{x}_i \in S}$
Estimate score of other points in N_f by interpolation
Compute $q_i^{(t)}$ according to Equation (8)
$M_f \leftarrow |M_f|$ points sampled with $q_i^{(t)}$ from N_f
Compute $\alpha_i^{(t)}$ according to Equation (6)
Compute $\alpha_i'^{(t)}$ according to Equation (9)
Compute $\mathrm{Sim}(\boldsymbol{v}^{(t_0)}, \boldsymbol{v}^{(t)})$ according to Equation (11)
if $\mathrm{Sim}(\boldsymbol{v}^{(t_0)}, \boldsymbol{v}^{(t)}) < \gamma$ **then**
    $g_i^{(t)} \propto \|q_i^{(t-1)} - q_i^{(t_0)}\|, \ i \in \{1, 2, \cdots, |N_f|\}$
    $q_i^{(t_0)} \leftarrow q_i^{(t)}, \ i \in \{1, 2, \cdots, |N_f|\}$
    $S \leftarrow |S|$ points sampled with g_i from N_f
    $t_0 \leftarrow t$
    Update mesh by Delaunay Triangulation
end if
return $M_f, \boldsymbol{\alpha}'$

select anchor points

estimate other points by interpolation

sample, calculate weight

update anchor points if necessary

Schrödinger equation, Viscous Burgers' equation and Korteweg-de Vries equation

PDE	Network Config		Optimizer Config	DMIS Config			Dataset Config		
	Depth	Width	Learning rate	$\|S\|$	γ	β	$\|N_f\|$	$\|N_i\|$	$\|N_b\|$
Schrödinger	4	64	0.001	1000	0.4	2	60000	200	200
Burgers	3	32	0.005	1000	0.4	1.5	100000	2000	2000
KdV	4	64	0.001	1000	0.4	2	60000	2000	2000

Method	Schrödinger Equation			Burgers' Equation			KdV Equation		
	ME	MAE	RMSE	ME	MAE	RMSE	ME	MAE	RMSE
PINN-O	1.360	0.186	0.4092	0.451	0.0738	0.1100	2.140	0.363	0.520
PINN-N	0.948	0.149	0.2906	0.358	0.0579	0.0859	1.860	0.292	0.441
xPINN	1.617	0.124	0.0704	0.317	0.0115	0.0014	2.486	0.244	0.194
cPINN	1.594	0.148	0.0783	0.399	0.0097	0.0012	2.760	0.270	0.249
PINN-DMIS(ours)	**0.647**	0.127	0.2196	**0.225**	0.0294	0.0495	**1.170**	0.391	0.492
xPINN-DMIS(ours)	0.999	0.046	0.0021	0.270	0.0103	0.0011	1.596	**0.202**	**0.128**
cPINN-DMIS(ours)	1.005	**0.043**	**0.0018**	0.258	**0.0088**	**0.0007**	2.788	0.238	0.210

	TC_2/s	TC_3/s	NC_2	NC_3
PINN-O	151	2005	4740	61984
PINN-N	466	5968	5681	72959
PINN-DMIS(ours)	**34**	**399**	**847**	**10219**

Table 3: Evaluation results of convergence behavior for solving the Schrödinger Equation

	TC_2/s	TC_3/s	NC_2	TC_3
PINN-O	**15**	97	1218	7308
PINN-N	83	417	1218	6032
PINN-DMIS(ours)	20	**89**	**1160**	**5336**

Table 4: Evaluation results of convergence behavior for solving the Viscous Burgers' Equation

	TC_2/s	TC_3/s	NC_2	NC_3
PINN-O	476	1812	12631	48401
PINN-N	794	4001	9855	49912
PINN-DMIS(ours)	**288**	**1046**	**6954**	**25029**

Table 7: Evaluation results of convergence behavior for solving the KdV Equation

$\|S\|$	γ	Schrödinger Equation			Burgers' Equation			KdV Equation		
		ME	MAE	RMSE	ME	MAE	RMSE	ME	MAE	RMSE
1000	0.2	0.762	0.120	0.233	0.593	0.039	0.085	1.460	0.376	0.526
1000	0.4	**0.647**	0.127	0.220	**0.225**	**0.029**	**0.049**	**1.170**	0.391	0.492
1000	0.6	0.838	**0.111**	**0.208**	0.919	0.057	0.147	1.760	**0.351**	0.518
10000	0.4	0.913	0.132	0.262	0.392	0.033	0.075	1.280	0.366	**0.475**
20000	0.4	1.100	0.163	0.332	0.620	0.046	0.105	1.730	0.452	0.600

Table 6: Ablation studies of the set size $|S|$ and the mesh update threshold γ for solving the Schrödinger Equation, the Viscous Burgers' Equation and the KdV Equation.