D3M : A deep domain decomposition method for solving PDEs

ShanghaiTech University

Ke Li
Collaborators: Kejun Tang, Tianfan Wu, Qifeng Liao

12th Annual meeting of China Society of Computational Mathematics

August 3, 2019

Outline

1. Problem setup
2. Deep learning for PDEs
3. Domain decomposition method
4. Deep domain decomposition method
5. Numerical study

Problem setup

Our goal is solving PDEs by using deep learning.
Time-independent PDEs:

$$\begin{aligned} \mathcal{L}(u(x);\alpha(x)) &= f(x), \quad x \in \Omega \\ u(x) &= g(x), \quad x \in \partial\Omega \end{aligned} \tag{1}$$

Time-dependent PDEs:

$$\begin{aligned} \partial_t u(t,x) + \mathcal{L}u(t,x) &= 0, \qquad (t,x) \in [0,T] \times \Omega \\ u(0,x) &= u_0(x), \qquad x \in \Omega \\ u(t,x) &= g(t,x), \qquad x \in [0,T] \times \partial\Omega \end{aligned} \tag{2}$$

Deep learning for PDEs

Deep Galerkin method (DGM) :
The Deep Galerkin method is proposed by J. Sirignano and K. Spiliopoulos in [4] for solving time-dependent PDEs (2).
The loss function of DGM is a simple MSE formula,

$$\begin{aligned} J(f;\theta) = & \left\| \frac{\partial f}{\partial t}(t,x;\theta) + \mathcal{L}f(t,x;\theta) \right\|^2_{[0,T]\times\Omega} + \|f(t,x;\theta) - g(t,x)\|^2_{[0,T]\times\partial\Omega} \\ & + \|f(0,x;\theta) - u_0(x)\|^2_{\Omega} \end{aligned}$$

So we minimize loss function $J(f;\theta)$, and the corresponding $f(t,x;\theta)$ is the approximation of objective function $u(t,x)$.
We notice that u does not exist in $J()$, so it does not require any training data which is the key to avoid "curse of dimensionality".

Deep Ritz method (DRM) :

Comparing with DGM, Deep Ritz method [5] uses variational formulation. A toy problem for example

$$\begin{cases} -\Delta u(x,y) = f & , \text{ in } \Omega, \\ u(x,y) = g(x,y) & , \text{ on } \partial\Omega, \end{cases} \tag{3}$$

The loss function with variational formula is

$$\begin{aligned} J(u) = \int_{\Omega} (\frac{1}{2}|\nabla u(x,y)|^2 - u(x,y)f(f,y))dxdy \\ + q\int_{\partial\Omega}(u(x,y)-g(x,y))^2 dxdy, \end{aligned} \tag{4}$$

where q is Lagrangian multiplier.

If we replace u by neural network N and minimize the loss function, the output of neural network can be regard as the output of function u.

Mixed residual loss

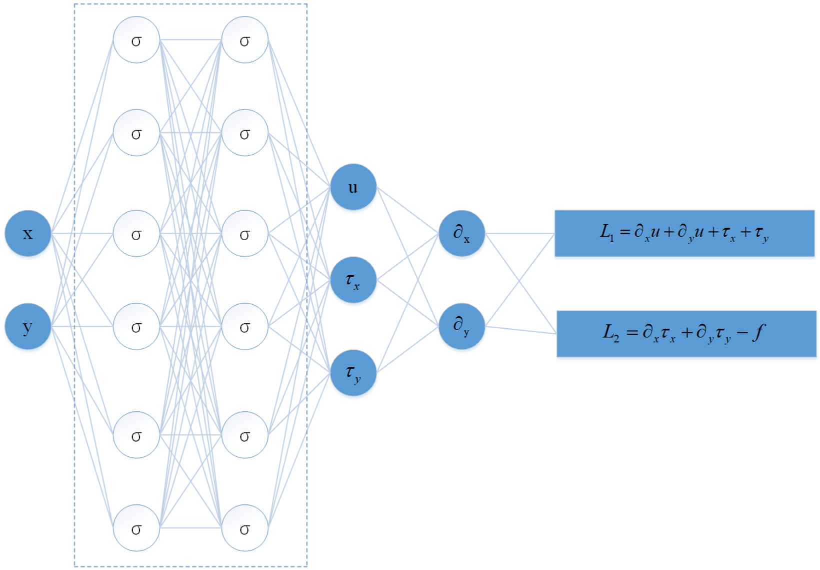
We employ a mixed residual loss [6] following Hellinger-Reissner principle [1]. With an additional variable $\tau \in \mathcal{H}(\mathbf{div})$, which represents flux, we can turn Equation (3) into

$$\begin{cases} \tau = -\nabla u & , \text{ in } \Omega, \\ \nabla \cdot \tau = f & , \text{ in } \Omega. \end{cases} \tag{5}$$

Mixed residual loss is

$$L(\tau, u, q) = \int_{\Omega} [(\tau + \nabla u)^2 + (\nabla \cdot \tau - f)^2] dxdy + q \int_{\partial\Omega} udxdy. \tag{6}$$

Corresponding neural network


σ
x
y
u
τ_x
τ_y
∂_x
∂_y
$L_1 = \partial_x u + \partial_y u + \tau_x + \tau_y$
$L_2 = \partial_x \tau_x + \partial_y \tau_y - f$

Domain decomposition method

Given a classical Poisson's equation

$$\begin{cases} -\Delta u = f & , \ \text{in } \Omega, \\ u = 0 & , \ \text{on } \partial\Omega. \end{cases} \tag{7}$$

We divide Ω into two overlapping subdomains $\Omega_i, i = 1, 2$ (see Figure 1), where

$$\Omega = \Omega_1 \cup \Omega_2,\ \Gamma_1 := \partial\Omega_1 \cap \Omega_2,\ \Gamma_2 := \partial\Omega_2 \cap \Omega_1,\ \Omega_{1,2} := \Omega_1 \cap \Omega_2. \tag{8}$$

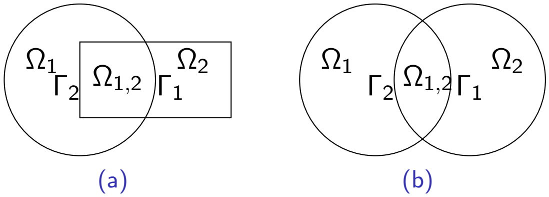


Figure: Partition into two overlapping subdomains.

We introduce the original form named Schwarz alternating method [3] here. Let u^0 be an initial guess defined in Ω and vanishing on $\partial\Omega$. For $k \geq 0$, we define sequences u_i^k where u_i^k denotes u^k in Ω_i. The u_i^{k+1} is determined from an iteration algorithm:

$$\begin{cases} -\Delta u_1^{k+1/2} = f & , \text{ in } \Omega_1, \\ u_1^{k+1/2} = u_2^k & , \text{ on } \Gamma_1, \\ u_1^{k+1/2} = 0 & , \text{ on } \partial\Omega_1 \cap \partial\Omega \end{cases} \tag{9}$$

and

$$\begin{cases} -\Delta u_2^{k+1} = f & , \text{ in } \Omega_2, \\ u_2^{k+1} = u_1^{k+1/2} & , \text{ on } \Gamma_2, \\ u_2^{k+1} = 0 & , \text{ on } \partial\Omega_2 \cap \partial\Omega. \end{cases} \tag{10}$$

Deep domain decomposition method (D3M)

We notice that normal deep variational networks and mixed residual networks perform worse with in-homogeneous boundary conditions, so that we define a function to overcome this weakness.

Definition

(Boundary function) A smooth function $\mathfrak{g}(x, y)$ *is called boundary function associated with* Ω *if*

$$\mathfrak{g}(x, y) = e^{-a \cdot d(x,y,\partial\Omega)} u(x, y), \quad (x, y) \in \Omega, \tag{11}$$

where $a \gg 1$ *is a coefficient, the notation* $d(x, y, \partial\Omega)$ *denotes the shortest Euclidean distance between* (x, y) *and* $\partial\Omega$*. If the point* (x, y) *is on the boundary,* $\mathfrak{g}(x, y) = u(x, y)$*. If not, the value of* $\mathfrak{g}(x, y)$ *will decrease to zero sharply.*

Let $v_i = u_i - \mathfrak{g}_i$ on each subdomain, and Equation can be represented as

$$\begin{cases} \tau = -\nabla v_i & , \text{ in } \Omega_i, \\ \nabla \cdot \tau = f + \Delta \mathfrak{g}_i & , \text{ in } \Omega_i. \end{cases} \tag{12}$$

Mixed residual loss is

$$L(\tau_i, v_i, q) = \int_{\Omega_i} [(\tau_i + \nabla v_i)^2 + (\nabla \cdot \tau_i - f - \Delta \mathfrak{g}_i)^2] dxdy + q \int_{\partial \Omega_i} v_i dxdy. \tag{13}$$

Algorithm 1 Deep domain decomposition

```
Input: $\Omega = [x_0, x_1] \times [y_0, y_1]$, $p$, $S_i$, $\Gamma_i$, $\eta$, $\theta$, $n$, $m_1$, $m_2$;
Initialize: $\epsilon = 10 \times \eta$, $k = 0$, $gv_i^0 = \mathbf{0}$, ;
Divide sparse domain into $\Omega_1, \cdots, \Omega_p$;
while $\epsilon > \eta$ do
    Run Algorithm (2) in each subdomain in parallel;
    $\epsilon = \frac{1}{p} \sum_{i=1}^{p} \|Sol_i^{(k+1)} - Sol_i^{(k)}\|_2^2$;
    $k = k + 1$;
end while
Merge $p$ parts $Sol_i^{(k)}$ and get $Dnn_{sol}^{(k)}$;
Return: $Dnn_{sol}^{(k)}$.
```

Algorithm 2 Training for subdomain Ω_i

```
Input: $gv_i^k$, $\Gamma_i$, $S_i$, $n$, $m_1$, $m_2$;
Construct function $\mathfrak{g}_i$ using value of $gv_i^k$, $v_i = \mathbf{N}_u - \mathfrak{g}_i$
for $n$ steps do
    Sample minibatch of $m_1$ samples $\hat{S}_i = \{(x_i, y_i)\}_{i=1}^{m_1}$ in $\Omega_i$;
    Sample minibatch of $m_2$ samples $g_i = \{(x_i, y_i)\}_{i=1}^{m_2}$ on $\partial\Omega_i$;
    Update the parameters $\theta_i$ by descending its stochastic gradient:
    $$\theta_i^{(k+1)} = \theta_i^{(k)} - \nabla_\theta \frac{1}{m_1} \sum_{i=1}^{m_1} [(\mathbf{N}_\tau^{(i)} + \nabla v^{(i)})^2 + (\nabla \cdot \mathbf{N}_\tau^{(i)} - f - \Delta \mathfrak{g}_i)^2] - q \nabla_\theta \frac{1}{m_2} \sum_{j=1}^{m_2} (v^{(j)})^2.$$
end for
$Sol_i^{(k+1)} = \mathbf{N}_u(S_i)$;
$gv_i^{(k+1)} = \mathbf{N}_u(\Gamma_i)$;
Return: $Sol_i^{(k+1)}$, $gv_i^{(k+1)}$;
```

The procedure of D3M is as follows. We first divide the domain Ω into d subdomains, and each two neighbor subdomains overlapping. The solution of a PDE on each local subdomain is replaced by a neural networks that can be trained through variational principle, while the solution on the whole domain consists of these solutions on each local domain. To be more precise, let Γ_i denote decomposed junctions, θ is initial weights of neural network, η is accuracy, N is number of samples generated in Ω_i to evaluate the output of network in each iteration, S_i is test samples in subdomain Ω_i, g_i is test samples on γ_i, n is training times in each iteration, m_1 and m_2 are batch sizes, $\mathbf{N_u}$ is the neural network for u, $\mathbf{N}_\tau$ is the neural network for τ , k is iteration times, and $Sol_i^{(k+1)}$ is network output for subdomain Ω_i in $k+1_{th}$ iteration. The formal description of D3M can be found in Algorithm 1.

D3M sampling

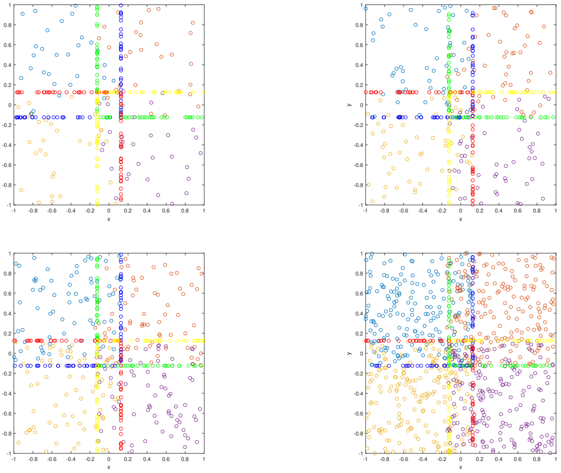


Figure: D3M sampling : a new type of mesh-free sampling.

Theorem

$J_i(\mathbf{N}_i)$ denotes the objective function on the subdomain Ω_i. Under above assumptions, for $\forall\epsilon > 0$, $\exists M > 0$, while iteration times $k > M$, $\mathbf{N}_i$ converges to optimal solution u_i^* of $J(u_i)$ in subdomain Ω_i for a constant $C > 0$

$$|\mathbf{N}_i - u_i^*|^2 \leq C\epsilon \ \text{ in } \Omega_i. \tag{14}$$

Theorem

For a given boundary function $\mathfrak{g}$ and a fixed q, the optimal solution $\mathbf{N}_u^*$ of Equation (6) and v^* of Equation (12) satisfy $\mathbf{N}_u^* = v^* + \mathfrak{g}$.

Then, we extend D3M to more general quasilinear parabolic PDEs (15) with physics-constrained approaches.

$$\begin{cases} \operatorname{div}(\alpha(x, u_i(x), \tau_i(x))) + \gamma(x, u_i(x), \tau_i(t, x)) = 0 & , \text{for } x \in \Omega_i \\ u_i(t, x) = 0 & , \text{for } x \in \partial\Omega_i \end{cases} \tag{15}$$

where τ_i denotes ∇u_i, and $\Omega_i \in \mathbb{R}^d$ are decomposed boundary sets with smooth boundaries $\partial\Omega_i$. We recall the network space of subdomain Ω_i with generated data according to [2]

$$\mathfrak{N}_i^n(\sigma) = \left\{ h(x) : \mathbb{R}^k \to \mathbb{R} | h(x) = \sum_{j=1}^{n} \beta_j \sigma \left(\alpha_j^T x - \theta_j \right) \right\} \tag{16}$$

where σ is any activation function, $\mathbf{x} \in \mathbb{R}^k$ is one set of generated data, $\beta \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^{k \times n}$ and $\theta \in \mathbb{R}^{k \times n}$ denote coefficients of networks.

Set $\mathfrak{N}_i(\sigma) = \bigcup_{n=1}^{\infty} \mathfrak{N}_i^n(\sigma)$. Under the universal approximation of neural networks, in each subdomain the neural networks f_i^n satisfies

$$\begin{cases} \operatorname{div}(\alpha(x, f_i^n(x), \tau_i^n(x))) + \gamma(x, f_i^n(x), \tau_i^n(t,x)) = h^n & , \text{for } x \in \Omega_i \\ f_i^n(t,x) = b^n & , \text{for } x \in \partial\Omega_i \end{cases} \tag{17}$$

where h^n and b^n satisfy

$$\|h^n\|_{2,\Omega_i}^2 + \|b^n\|_{2,\partial\Omega_i}^2 \to 0, \quad as \quad n \to \infty. \tag{18}$$

Theorem

Suppose the domain Ω is decomposed into $\{\Omega_i\}_{i=1}^{p}$, $k > 0$ denotes iteration times (omitted in notations for brief). $\mathfrak{N}_i(\psi)$ denotes networks space space in subdomain Ω_i, where subdomains are compact. Assume that target function (15) *has unique solution in each subdomain, nonlinear terms $div(x, u, \nabla u)$ and $\gamma(x, u, \nabla u)$ are locally Lipschitz in $(u_i, \nabla u_i)$, and ∇u_i^k uniformly converges to ∇u_i^* with k. For $\forall\ \epsilon > 0$, there $\exists\ K > 0$ such that there exists a set of neural networks $\{\mathbf{N}_i \in \mathfrak{N}_i(\psi)\}_{i=1}^{p}$ satisfies the L^2 error $E_2(\mathbf{N}_i)$ as follow*

$$\sum_{i=1}^{p} \lim_{k\to\infty} E_2(\mathbf{N}_i) \leq K\epsilon^2. \tag{19}$$

Theorem

Under Assumptions and Equation (18)*, with iteration times $k \to \infty$, the set of neural networks $\mathbf{N}_i$ converge to the unique solutions to* (15)*, strongly in $\mathcal{L}^{\rho}(\Omega_i)$ for every $\rho < 2$. In addition, in each subdomain the sequence $\{\mathbf{N}_i^n(x)\}_{n\in\mathbb{N}}$ is bounded in n under the constraint of* Proposition *2 and converges to u_i.*

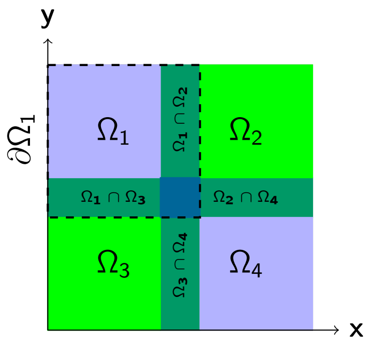
Numerical test

Poisson's equation

$$\begin{cases} -\Delta u(x,y) = 1 & , \ \text{in } \Omega, \\ u(x,y) = 0 & , \ \text{on } \partial\Omega, \end{cases} \tag{20}$$

where the spatial domain $\Omega = [-1,1] \times [-1,1]$.

In order to show the dimensions, we re-plot this spatial domain in Figure 3a

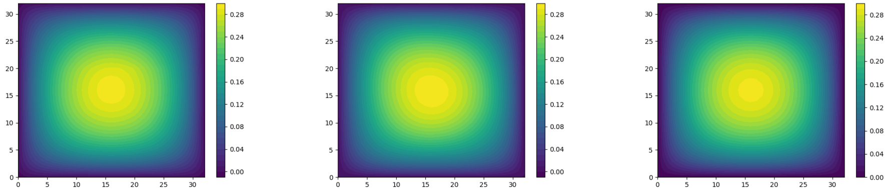


(a) Spatial domain with four overlapping components.

Figure: Illustrations of the spatial domain.

Results

The results from D3M is shown in Figure 4. For comparison, we plot the result of normal Deep Ritz method (DRM) with the same type of network and finite element method (FEM), see Figure 4b and 4c. We set the result from FEM as the ground truth and relative error $e_r = \frac{\|sol - fem_{sol}\|_2}{\|fem_{sol}\|_2}$. We compare results using Residual network, and the comparison including relative error and max error can be found in Table 1. We can see that our D3M offers a higher accuracy than normal DRM.



(a) Solution of D3M (b) Solution of DRM (c) Solution of FEM

Figure: Solutions computed by three different approaches.

Table: The relative error and max error.

Method	Net type	Layers	Parameters	Relative error
DRM	Resnet	4	2048	0.0271
DRM	Resnet	8	4096	0.0157
D3M	Resnet	4	2048	0.0065
D3M	Resnet	8	4096	0.0045

Schrödinger equation :

$$\left[\frac{-\bar{h}^2}{2m}\nabla^2 + V(\mathbf{r})\right]\Psi(\mathbf{r}) = E\Psi(\mathbf{r}). \tag{21}$$

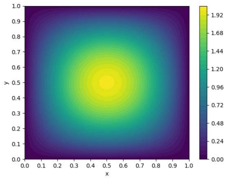
Where $\bar{h} = \frac{h}{2\pi}$ is the reduced Planck constant, m is the particle's mass and E is a known constant related to the energy level. This equation occurs often in quantum mechanics where $V(\mathbf{r})$ is the function for potential energy. Here we consider an infinite potential well

$$V(\mathbf{r}) = \begin{cases} 0, & \mathbf{r} \in [0,1]^d \\ \infty, & \mathbf{r} \notin [0,1]^d. \end{cases} \tag{22}$$

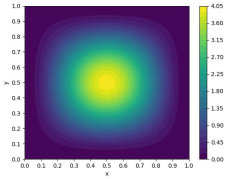
The residual loss is

$$\begin{aligned} L(\tau_i, \mathbf{N}_i, q) = & \int_{\Omega_i} [\Delta \mathbf{N}_i(r)) + \Delta \mathfrak{g}_i - E\mathbf{N}_i(r)) + E\mathfrak{g}_i] dr \\ & + q\int_{\partial\Omega_i} (\mathbf{N}_i(r))^2 dr + \gamma(\frac{\Omega_i}{\Omega}\int_{\Omega_i} |\mathbf{N}_i(r)|^2 dr - 1 + P_i)^2, \end{aligned} \tag{23}$$

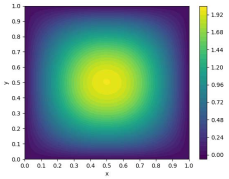
where $\int_\Omega |\Psi(r)|^2 dr = 1$, because $\Psi(r)$ is wave function and $\Psi(r)^2$ means probability density of particle appearing. $\frac{\Omega_i}{\Omega}$ is the ratio of domain's area and P_i denotes the probability of particle appearing in $\Omega \backslash \Omega_i$.



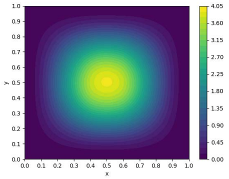
(a) Groundtruth of $\Phi(r)$



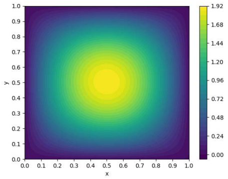
(b) Groundtruth of $\Phi^2(r)$



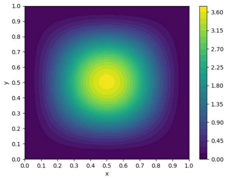
(c) D3M solution of $\Phi(r)$



(d) D3M solution of $\Phi^2(r)$

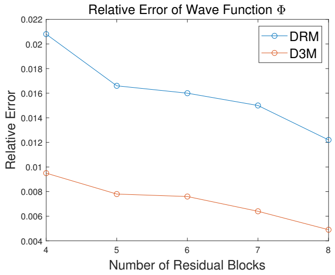


(e) DRM solution of $\Phi(r)$



(f) DRM solution of $\Phi^2(r)$

Figure: The solutions of wave function and probability density for a time-independent Schrödinger equation.



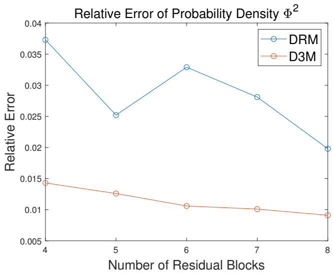


Figure: Comparison of relative error with different number of residual blocks.

Table: The relative error for both wave function and probability density.

Target	Method	Net type	Blocks	Number of neurons	Relative error
Wave	DRM	ResNet	4	2048	0.0209
Wave	DRM	ResNet	8	4096	0.0169
Wave	D3M	ResNet	4	2048	0.0095
Wave	D3M	ResNet	8	4096	0.0045
Prob.	DRM	ResNet	4	2048	0.0357
Prob.	DRM	ResNet	8	4096	0.0334
Prob.	D3M	ResNet	4	2048	0.0143
Prob.	D3M	ResNet	8	4096	0.0091

Thank You!

Douglas N. Arnold.
Mixed finite element methods for elliptic problems.
Computer Methods in Applied Mechanics and Engineering, 82(1):281 – 300, 1990.

Kurt Hornik.
Approximation capabilities of multilayer feedforward networks.
Neural networks, 4(2):251–257, 1991.

Hermann Amandus Schwarz.
Ueber einen Grenzübergang durch alternirendes Verfahren.
Zürcher u. Furrer, 1870.

Justin Sirignano and Konstantinos Spiliopoulos.
Dgm: A deep learning algorithm for solving partial differential equations.
Journal of Computational Physics, 375:1339–1364, 2018.

E Weinan and Bing Yu.

The deep ritz method: A deep learning-based numerical algorithm for solving variational problems.
Communications in Mathematics and Statistics, 6(1):1–12, 2018.

Yinhao Zhu, Nicholas Zabaras, P Koutsourelakis, and Paris Perdikaris.
Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data.
arXiv preprint arXiv:1901.06314, 2019.