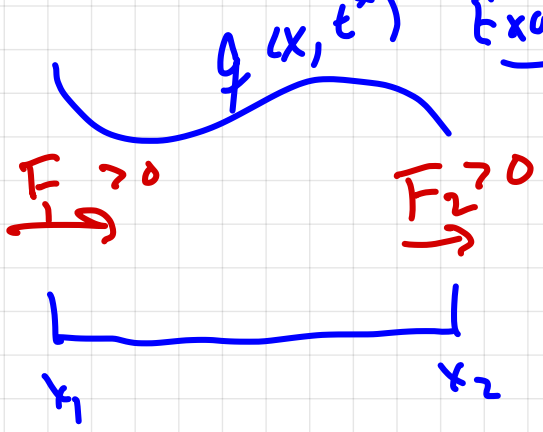
Conservation laws

$$\underline{u}_t + \nabla \cdot F(\underline{u}) = 0$$

$\underline{u}$ = some conserved quantity



Example

let $q(x,t)$ be a concentration

Then $\int_{x_1}^{x_2} q(x,t)\,dx$ is

total mass at time t.

Mass can only change by a flux at the ends:

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)\,dx = F_1(t) - F_2(t)$$

Fluid? $F(q,x,t) = u(x,t) \cdot q(x,t)$

velocity · density

$\frac{m}{s} \cdot \frac{kg}{m}$

For constant velocity:

$$F = \bar{u}\, q(x,t) = F(q)$$

$$\rightarrow \frac{d}{dt}\int_{x_1}^{x_2} q(x,t)\,dx = f(q(x,t)) - f(q(x_2,t))$$

$$= -f(q(x,t))\Big|_{x_1}^{x_2}$$

$$= -\int_{x_1}^{x_2} \frac{d}{dx} f(q(x,t))\,dx$$

$$\rightarrow \int_{x_1}^{x_2} \frac{d}{dt} q(x,t) + \frac{\partial f(q(x,t))}{\partial x}\,dx = 0$$

other example:

ρ = density
u = velocity
(p = pressure)
E = energy

$$\begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix} + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (E+p)u \end{bmatrix}_x = 0$$

PINN

Main challenges:

1. nothing really PDE specific
   → how do we enforce conservation properties

2. cost
   → very monolithic at the point

Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems

Ameya D. Jagtap[a], Ehsan Kharazmi[a], George Em Karniadakis[a,b,*]

2020

→ CPINN

↓ also

physics constrained NN

Extended physics-informed neural networks (XPINNs) : A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations

Ameya D. Jagtap* [1]
George Em Karniadakis [1]

2021

→ XPINN

↓*

hp-PINN (variational)

D3M: A Deep Domain Decomposition Method for Partial Differential Equations

KE LI[1,2,3], **(Student Member, IEEE), KEJUN TANG**[1,2,3], **TIANFAN WU**[4], **AND QIFENG LIAO**[1]

2019

→ D3M

Parallel Physics-Informed Neural Networks via Domain Decomposition

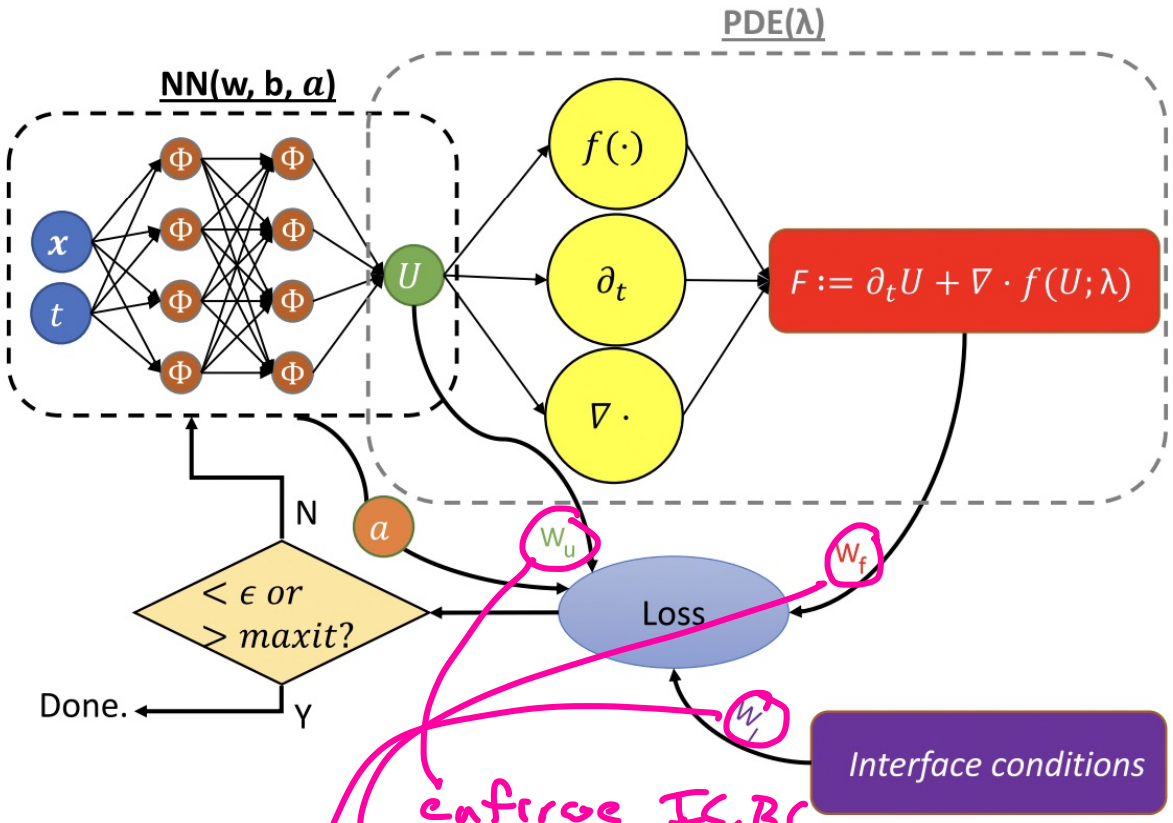
Khemraj Shukla, Ameya D. Jagtap, George Em Karniadakis*

→ PINN in parallel?

cPINN

Gist:

- use domain decomp to enforce solution locally.
- provide information on fluxes at interfaces
- per-domain selection of
  - depth
  - width
  - activation
  - etc.
- adaptive activations



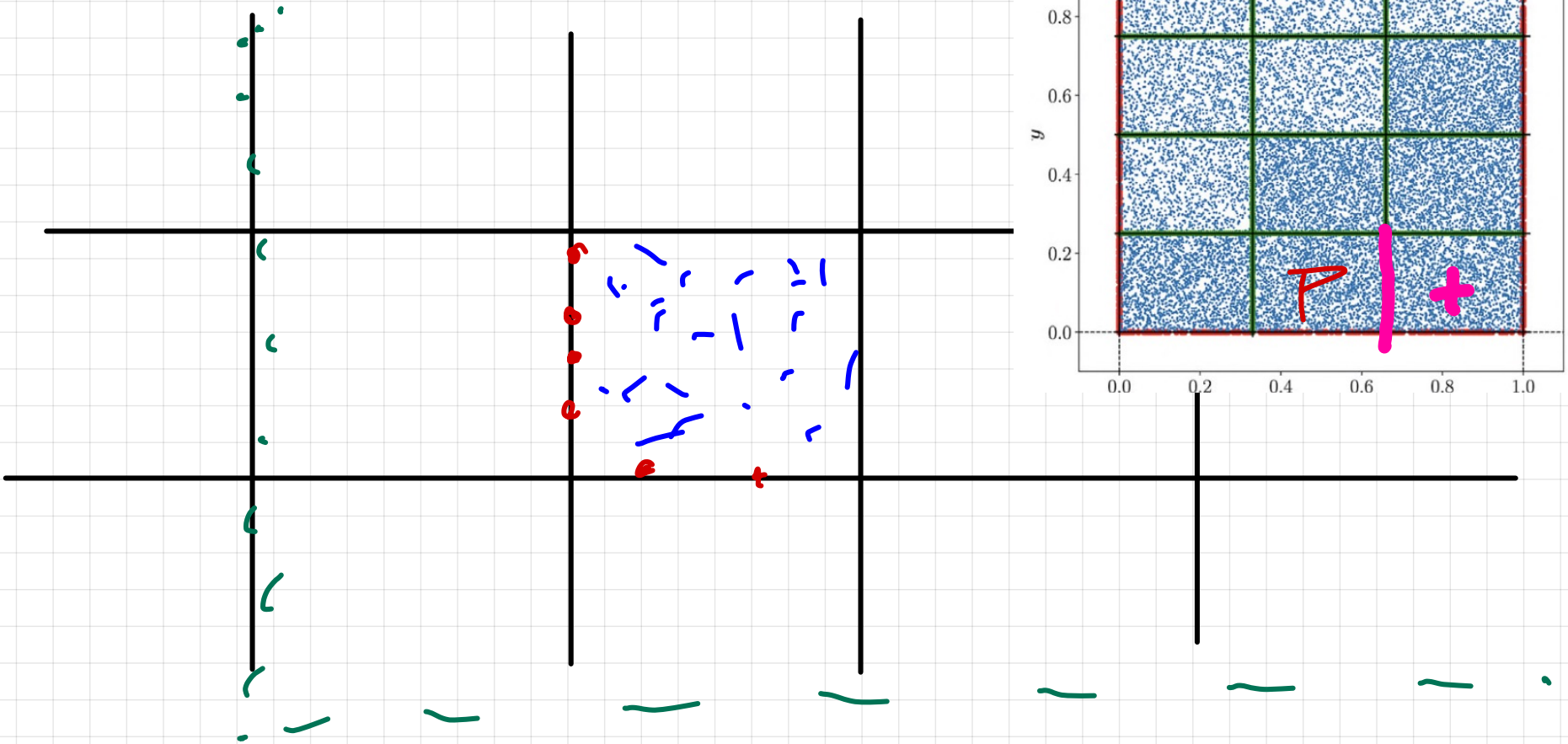
enforce IC, BC
enforce residual (collocation)
enforce flux continuity
+average solution in two neighboring NN.

let $\underline{x}_u$ = training pts

$\underline{x}_F$ = residual pts

x_I = interface pts.

loss = per sub domain



p^{th} domain:

$u - \{u\}$

$$\mathcal{L} = \frac{w_p}{N_p} \sum \left| u^i - u(x_p^i) \right|^2$$

$$+ \frac{w_F}{N_F} \sum \left| F(x_F^i) \right|^2$$

$$+ \frac{w_I}{N_I} \sum \left| f(u(x_I^i)) \cdot n - f_+(u(x_I^i)) \cdot n \right|^2$$

$$+ \frac{w_I}{N_I} \sum \left| u(x_I^i) - \frac{u(x_I^i) + u_+(x_I^i)}{2} \right|^2$$

$$\sum \frac{1}{q} \left(u(x_I^i) - u_+(u_I^i) \right)^2$$

t

Algorithm 1: cPINN algorithm

Step 1 : Specify the training set over all sub-domains

Training data : $\mathbf{u}_{\tilde{\Theta}_p}$ network $\{\mathbf{x}^i_{u_p}\}_{i=1}^{N_{up}}, \quad p = 1, 2, \cdots, N_{sd}.$

Residual training points : $\mathcal{F}$ network $\{\mathbf{x}^i_{F_p}\}_{i=1}^{N_{Fp}}, \quad p = 1, 2, \cdots, N_{sd}.$

Step 2 : Specify the interface points

Interface points : $\{\mathbf{x}^i_{I_p}\}_{i=1}^{N_{Ip}}$.

Step 3: Assign the common interface points to the neighbouring sub-domains.

Step 4 : Construct the neural network $\mathbf{u}_{\tilde{\Theta}_p}$ with random initialization of parameters $\tilde{\Theta}_p$ in each sub-domain.

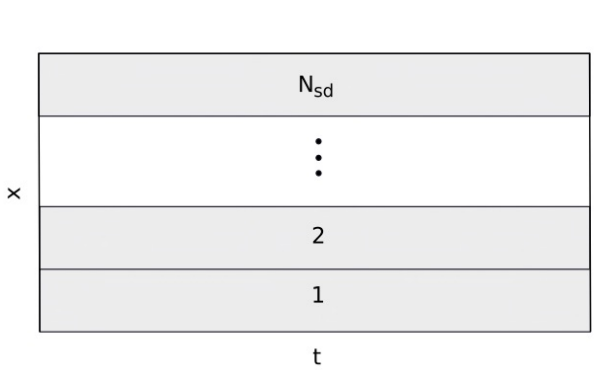
Step 5 : Construct the residual neural network $\mathcal{F}$ in each sub-domain by substituting surrogate $\mathbf{u}_{\tilde{\Theta}_p}$ into the governing equations using automatic differentiation and other arithmetic operations.

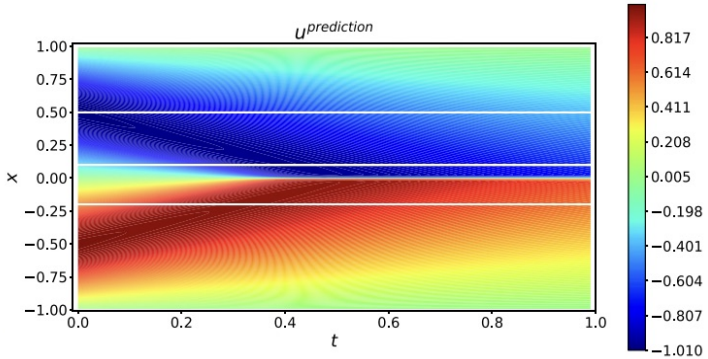
Step 6: Specify the loss function in the p^{th} sub-domain as

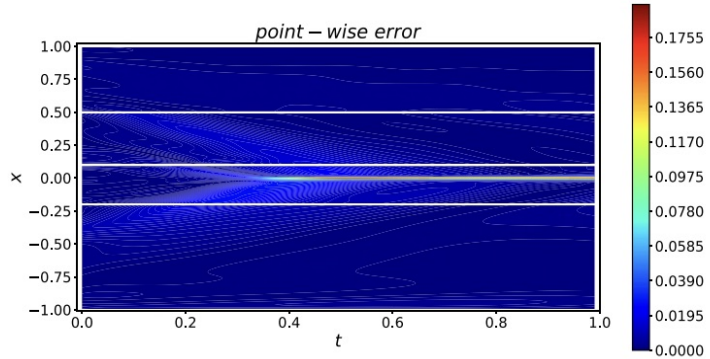
$$\mathcal{L}(\tilde{\Theta}_p) = \frac{W_{up}}{N_{up}} \sum_{i=1}^{N_{up}} \left| u^i - u(\mathbf{x}^i_{u_p}) \right|^2 + \frac{W_{\mathcal{F}_p}}{N_{F_p}} \sum_{i=1}^{N_{Fp}} \left| \mathcal{F}(\mathbf{x}^i_{F_p}) \right|^2 + W_{I_p} \times \text{Interface Conditions}, \quad p = 1, 2, \cdots, N_{sd}.$$

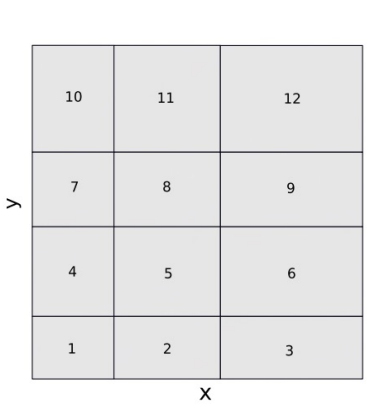
Step 7: Find the best parameters using suitable optimization method for minimizing the loss function in each sub-domain

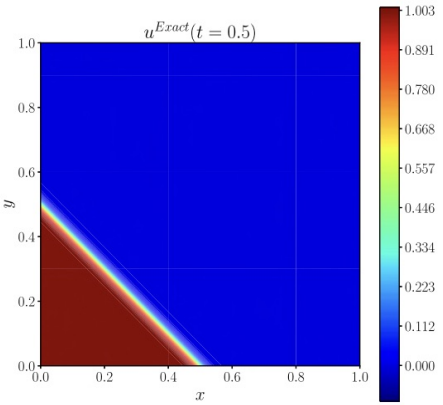
$$\tilde{\Theta}^*_p = \arg\min_{\tilde{\Theta}_p} (\mathcal{L}(\tilde{\Theta}_p)), \quad p = 1, 2, \cdots, N_{sd}. \tag{5}$$

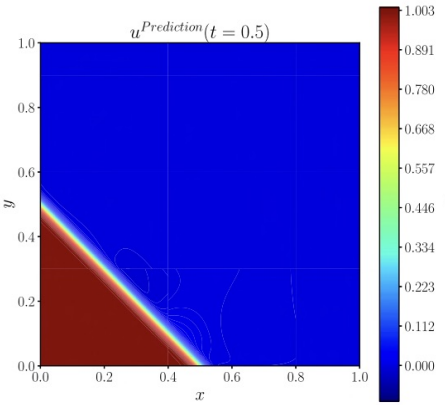

N_{sd}
2
1
x
t

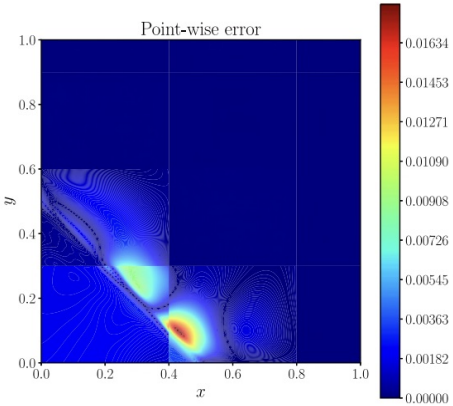

$u^{prediction}$
x
t


$point-wise\ error$
x
t


10
11
12
7
8
9
4
5
6
1
2
3
y
x

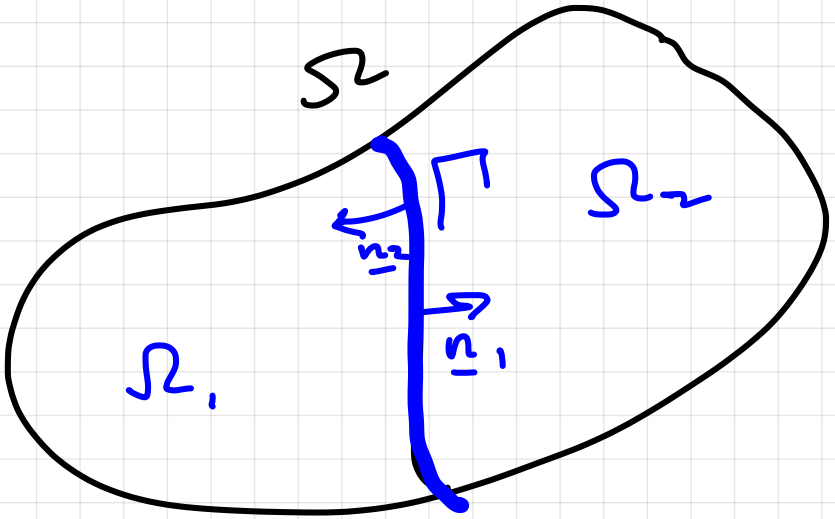

$u^{Exact}(t=0.5)$
y
x


$u^{Prediction}(t=0.5)$
y
x


Point-wise error
y
x

Back to D.D.

What is domain decomp?



Look at $-\nabla \cdot \nabla u = f$

$u = 0$ on $\partial\Omega$

Then $-\nabla \cdot \nabla u_1 = f$ on Ω_1

$u = 0$ on $\partial\Omega_1 \setminus \Gamma$

$-\nabla \cdot \nabla u_2 = f$ on Ω_2

$u = 0$ on $\partial\Omega_2 \setminus \Gamma$

$$\begin{cases} u_1 = u_2 \\ \underline{n}_1 \cdot \nabla u_1 = -n_2 \cdot \nabla u_2 \end{cases} \quad \text{on } \Gamma \qquad \text{(if smooth)}$$

D3M: A Deep Domain Decomposition Method for Partial Differential Equations

2019 use energy form

KE LI[1,2,3], (Student Member, IEEE), KEJUN TANG[1,2,3], TIANFAN WU[4], AND QIFENG LIAO[1]

Deep Domain Decomposition Method: Elliptic Problems

Wuyang Li LIWY648@NENU.EDU.CN
Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing 100194, China.
School of Mathematics and Statistics, Northeast Normal University, Changchun 130012, China.

Xueshuang Xiang* XIANGXUESHUANG@QXSLAB.CN
Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing 100194, China.

Yingxiang Xu YXXU@NENU.EDU.CN
School of Mathematics and Statistics, Northeast Normal University, Changchun 130012, China.

2020 energy form

Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations

Ameya D. Jagtap[1,*] and George Em Karniadakis[1,2]

[1] *Division of Applied Mathematics, Brown University, 182 George Street, Providence, RI 02912, USA.*
[2] *Pacific Northwest National Laboratory, Richland, WA 99354, USA.*

Received 26 August 2020; Accepted (in revised version) 3 October 2020

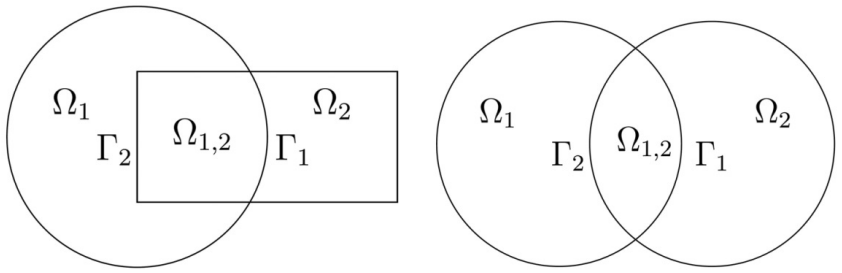
2020 not variational.

Combining machine learning and domain decomposition methods for the solution of partial differential equations—A review

2021 overview

Alexander Heinlein[1,2] | Axel Klawonn[1,2] | Martin Lanser[1,2] | Janine Weber[1]

DDM



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

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

let $\tau = -\nabla u$

$\nabla \cdot \tau = f$

$$\text{Loss} = \int (q + \nabla u)^2 + (\nabla \cdot q - f)^2$$

$$+ \quad \eta \int_{\partial\Omega} u \, d$$

η: Lagrange multiplier

$$\min \int_\Omega \frac{1}{2} |\nabla u|^2 \, dx - \int_\Omega f u \, dx \qquad \text{s.t. } u = 0 \text{ on } \partial\Omega.$$

$$\rightarrow \text{Lagrangian: } L(u, \eta) = \int \frac{1}{2} |\nabla u|^2 - \int u f \, dx$$

$$+ \eta \int_{\partial\Omega} u \, dx$$

Observation:

if u satisfies $\begin{aligned} -\Delta u &= f \\ u &= 0 \end{aligned}$

$\frac{1}{2}a(u,u) - \ell(u)$

$\frac{1}{2}(\nabla u, \nabla u) - (f,u)$

then u satisfies

$$\begin{cases} u \leftarrow \min_{u} \dfrac{\int_\Omega \frac{1}{2}|\nabla u|^2 dx - \int_\Omega f u \, dx}{\frac{1}{2}(\nabla u, \nabla u)} \\ \text{with } u = 0 \text{ on } \partial\Omega \end{cases}$$

$\rightarrow$ Lagrangian $L(u,q) = \int_\Omega \frac{1}{2}|\nabla u|^2 - \int_\Omega u f \, dx + q \int_\Omega u \, dx$

q = multiplier

if $\underline{\tau} = -\nabla u$

$\nabla \cdot \underline{\tau} = f$

thm

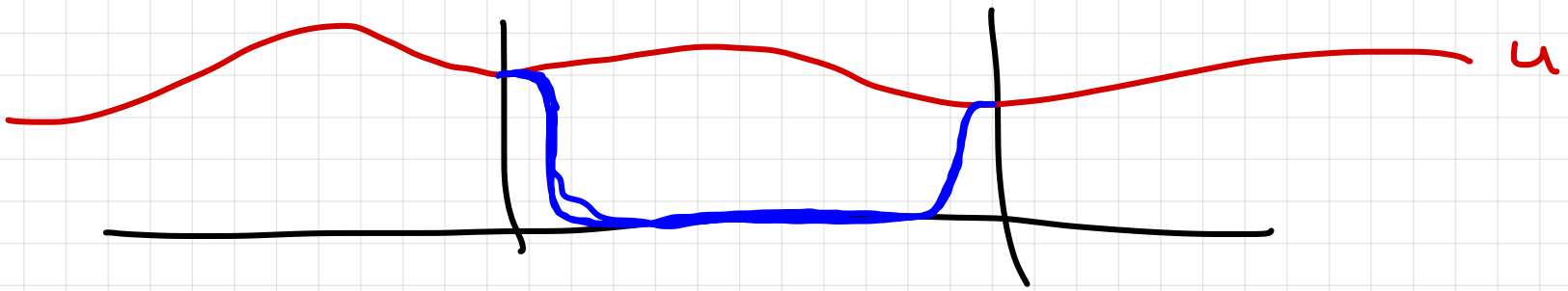
$$\mathcal{L}(\underline{\tau}, u, \gamma) = \int_\Omega (\underline{\tau} + \nabla u)^2 + (\nabla \cdot \tau - f)^2 \, dx$$

$$+ \gamma \int_{\partial\Omega} u \, dx$$

Let $N_u(x, \theta_u)$ be a NN model for u

$N_\tau(x, \theta_\tau)$ .. . ... $\underline{\tau}$.

Next piece



Ω_i

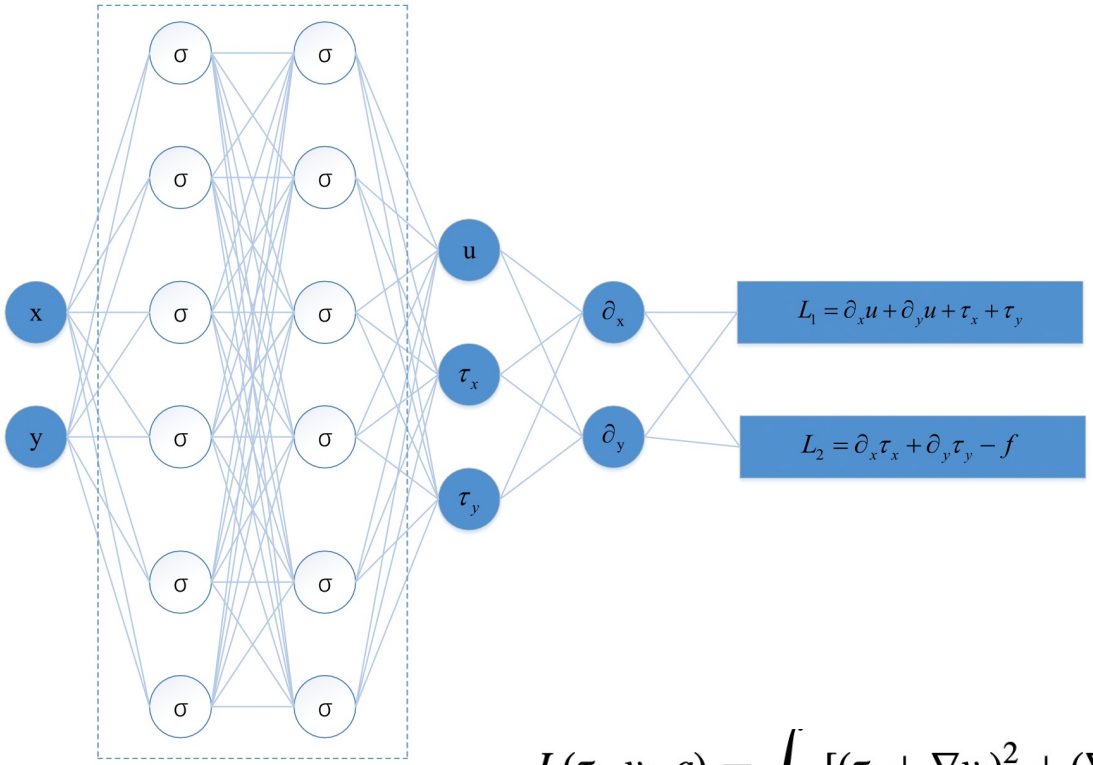
Let $g(x) = e^{-a\, d(x)}$ $u(x)$

$d(x)$: distance to $\partial\Omega_i$.

then let $v_i = u_i - g_i$

$$\Rightarrow \quad -\Delta v_i = f + \Delta g_i$$

$$v_i = 0 \quad \text{on } \partial\Omega_i$$



$$
\begin{aligned}
L(\tau_i, v_i, q) = & \int_{\Omega_i} [(\tau_i + \nabla v_i)^2 + (\nabla \cdot \tau_i - f - \Delta \mathfrak{g}_i)^2] dxdy \\
& + q \int_{\partial \Omega_i} v_i^2 \, dxdy, \\
\approx & \frac{1}{m_1} \sum_{k=1}^{m_1} [(\tau_i^{(k)} + \nabla v_i^{(k)})^2 + (\nabla \cdot \tau_i^{(k)} - f - \Delta \mathfrak{g}_i)^2] \\
& + \frac{1}{m_2} \sum_{j=1}^{m_2} q \cdot (v_i^{(j)})^2. \qquad (15)
\end{aligned}
$$

$$
\begin{aligned}
L(\tau_i, v_i, q) = & \int_{\Omega_i} [(\tau_i + \nabla v_i)^2 + (\nabla \cdot \tau_i - f - \Delta \mathfrak{g}_i)^2] dxdy \\
& + q \int_{\partial \Omega_i} v_i^2 \, dxdy, \\
\approx & \frac{1}{m_1} \sum_{k=1}^{m_1} [(\tau_i^{(k)} + \nabla v_i^{(k)})^2 + (\nabla \cdot \tau_i^{(k)} - f - \Delta \mathfrak{g}_i)^2] \\
& + \frac{1}{m_2} \sum_{j=1}^{m_2} q \cdot (v_i^{(j)})^2. \qquad (15)
\end{aligned}
$$

Algorithm 2 Training for Subdomain Ω_i

1: **Input:** S_i, gv_i^k, g_i^k, n, m_1, m_2.
2: Construct function $\mathfrak{g}_i$ using value of gv_i^k, $v_i = \mathbf{N}_u - \mathfrak{g}_i$.
3: **for** n steps **do**
4: Sample minibatch of m_1 samples $\hat{S}_i = \{(x_i, y_i)\}_{i=1}^{m_1}$ in Ω_i.
5: Sample minibatch of m_2 samples $\hat{g}_i = \{(x_i, y_i)\}_{i=1}^{m_2}$ on $\partial \Omega_i$.
6: Update the parameters θ_i by descending its stochastic gradient:

$$
\begin{aligned}
\theta_i^{(k+1)} = \theta_i^{(k)} - \nabla_\theta \frac{1}{m_1} \sum_{k=1}^{m_1} & [(\mathbf{N}_\tau^{(k)} + \nabla v_i^{(k)})^2 \\
& + (\nabla \cdot \mathbf{N}_\tau^{(k)} - f - \Delta \mathfrak{g}_i)^2] - \nabla_\theta \frac{1}{m_2} \sum_{j=1}^{m_2} (q \cdot v_i^{(j)})^2.
\end{aligned}
$$

7: **end for**
8: $Sol_i^{(k+1)} = \mathbf{N}_u(S_i)$.
9: $gv_i^{(k+1)} = \mathbf{N}_u(g_i^{(k)})$.
10: **Return:** $Sol_i^{(k+1)}$, $gv_i^{(k+1)}$.

Overview

$$\mathcal{L}u = f \quad \text{in } \Omega$$

$$u = g \quad \text{on } \partial\Omega$$

PINN: $L(\theta) = L_r(\theta) + L_{br}(\theta)$

$$\frac{1}{N_f} \sum \left| \mathcal{L}\left(N(x_f^i, \theta)\right) - f(x_f^i) \right|^2$$

$$\frac{1}{N_g} \sum \left| N(x_j^i, \theta) - g(x_j^i) \right|^2$$

Reformulate:

equivalent to

$$\min \int_\Omega \tfrac{1}{2} |\nabla u|^2 - fu \, dx$$

$$\text{s.t. } u = g \text{ on } \partial\Omega$$

Let $N(x, \theta)$ approximate u.

$$\Rightarrow \quad \tfrac{1}{2} |\nabla N(x, \theta)|^2 - f(x) N(x, \theta)$$

is a fcn of "x".

$$\Rightarrow \text{loss} = \int_\Omega \tfrac{1}{2} |\nabla N(x, \theta)|^2 - f(x) N(x, \theta)$$

use quadrature

$$+ q \int_{\partial\Omega} |N(x, \theta) - g(x)|^2 \, dx$$

Algorithm 1. Brief sketch of the DeepDDM and D3M algorithms; see [54, Algorithm 2] and [52, Algorithms 1 and 2], respectively, for details

Init: Weights and biases for all local PINNs (one for each subdomain) and interface solutions u_{Γ_s}, $s = 1, \dots, N$ for all collocation points on the local interface

Loop until convergence of the DD method

Train all local PINNs $\mathcal{N}_s$, $s = 1, \dots, N$, for all subdomains (parallelizable)

Communicate the Dirichlet data $\mathcal{D}(\mathcal{N}_s)$, $s = 1, \dots, N$, between neighboring subdomains

Update u_{Γ_s}, $s = 1, \dots, N$, using $\mathcal{D}(\mathcal{N}_r)$ obtained from the neighboring subdomains

End Loop

Back to XPINN

- NN in subdomain q is

$$u_{\tilde{\theta}_q}(z) = N(z, \tilde{\theta}_q)$$

- final solution is

$$u_\theta(z) = \sum_{q=1}^{\#} u_{\theta_q}(z) \cdot \mathbb{1}_{\Omega_q}$$

$$\mathbb{1} = \begin{cases} 0 & \text{if } z \notin \Omega_q \\ 1 & \text{if } z \in \Omega_q^0 \\ \gamma_s & \text{if } z \in \partial\Omega_q \end{cases}$$

$$\mathcal{J}(\tilde{\Theta}_q) = W_{u_q}\,\mathrm{MSE}_{u_q}(\tilde{\Theta}_q;\{\mathbf{x}_{u_q}^{(i)}\}_{i=1}^{N_{u_q}}) + W_{\mathcal{F}_q}\,\mathrm{MSE}_{\mathcal{F}_q}(\tilde{\Theta}_q;\{\mathbf{x}_{F_q}^{(i)}\}_{i=1}^{N_{F_q}})$$
$$+ W_{I_q}\underbrace{\mathrm{MSE}_{u_{avg}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}})}_{\text{Interface condition}} + W_{I_{\mathcal{F}_q}}\underbrace{\mathrm{MSE}_{\mathcal{R}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}})}_{\text{Interface condition}}$$
$$+\underbrace{\text{Additional Interface Condition's}}_{\text{Optional}},$$

$$\mathrm{MSE}_{u_q}(\tilde{\Theta}_q;\{\mathbf{x}_{u_q}^{(i)}\}_{i=1}^{N_{u_q}}) = \frac{1}{N_{u_q}}\sum_{i=1}^{N_{u_q}}\left|u^{(i)} - u_{\tilde{\Theta}_q}(\mathbf{x}_{u_q}^{(i)})\right|^2,$$

$$\mathrm{MSE}_{\mathcal{F}_q}(\tilde{\Theta}_q;\{\mathbf{x}_{F_q}^{(i)}\}_{i=1}^{N_{F_q}}) = \frac{1}{N_{F_q}}\sum_{i=1}^{N_{F_q}}\left|\mathcal{F}_{\tilde{\Theta}_q}(\mathbf{x}_{F_q}^{(i)})\right|^2,$$

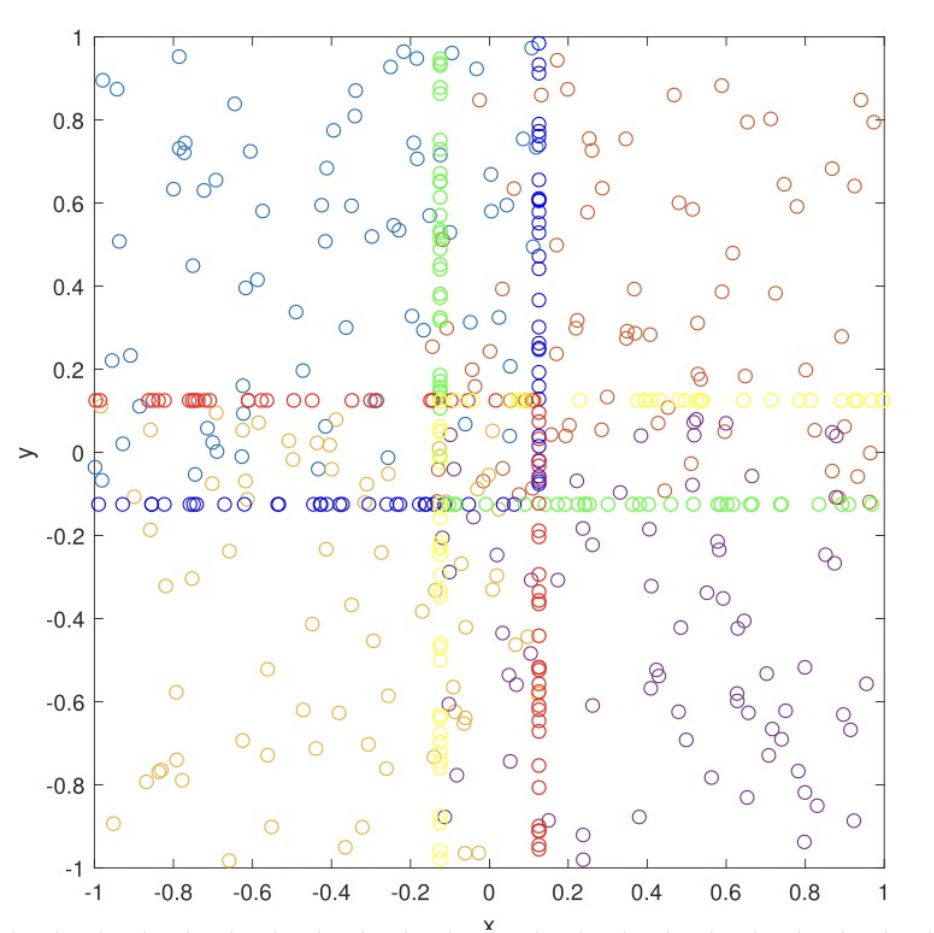
PINN

$$\mathrm{MSE}_{u_{avg}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}}) = \sum_{\forall q^+}\left(\frac{1}{N_{I_q}}\sum_{i=1}^{N_{I_q}}\left|u_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)}) - \left\{\left\{u_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)})\right\}\right\}\right|^2\right),$$

- enforce average

$$\mathrm{MSE}_{\mathcal{R}}(\tilde{\Theta}_q;\{\mathbf{x}_{I_q}^{(i)}\}_{i=1}^{N_{I_q}}) = \sum_{\forall q^+}\left(\frac{1}{N_{I_q}}\sum_{i=1}^{N_{I_q}}\left|\mathcal{F}_{\tilde{\Theta}_q}(\mathbf{x}_{I_q}^{(i)}) - \mathcal{F}_{\tilde{\Theta}_{q^+}}(\mathbf{x}_{I_q}^{(i)})\right|^2\right),$$

- enforce residual continuity.


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
x

Today:
- DD for elliptic
- inverse problems

Wed:
- code for DD + inverse
- @ NCSA ?

M 10/2:
- project ideas
- Neural ops ?

W 10/4:
- project outline, roadmaps.