Numerical linear algebra with neural operator preconditioning for solving some parametric PDEs

27th Conference of the International Linear Algebra Society (ILAS 2026)

May 18, 2026, Virginia Tech, Blacksburg, USA

Yanfei Xiang (✉ yanfei.xiang@math.unistra.fr)

https://irma.math.unistra.fr/∼xiang/

MOCO team, IRMA, University of Strasbourg; MACARON team, Inria

IRMA

Some parameteric PDEs from dynamic systems

Parametric Poisson equation (a common elliptic equation):

$$\nabla^2 u(x) = \rho(x) \tag{1}$$

Heterogeneous Darcy flow (groundwater flow through porous media):

$$-\nabla\big(p(x)\nabla u(x)\big) = \rho(x) \tag{2}$$

Heterogeneous Convection-Diffusion equation:

$$-\nabla\big(d(x)\nabla u(x)\big) + \vec{c}(x)\nabla u(x) = \rho(x) \tag{3}$$

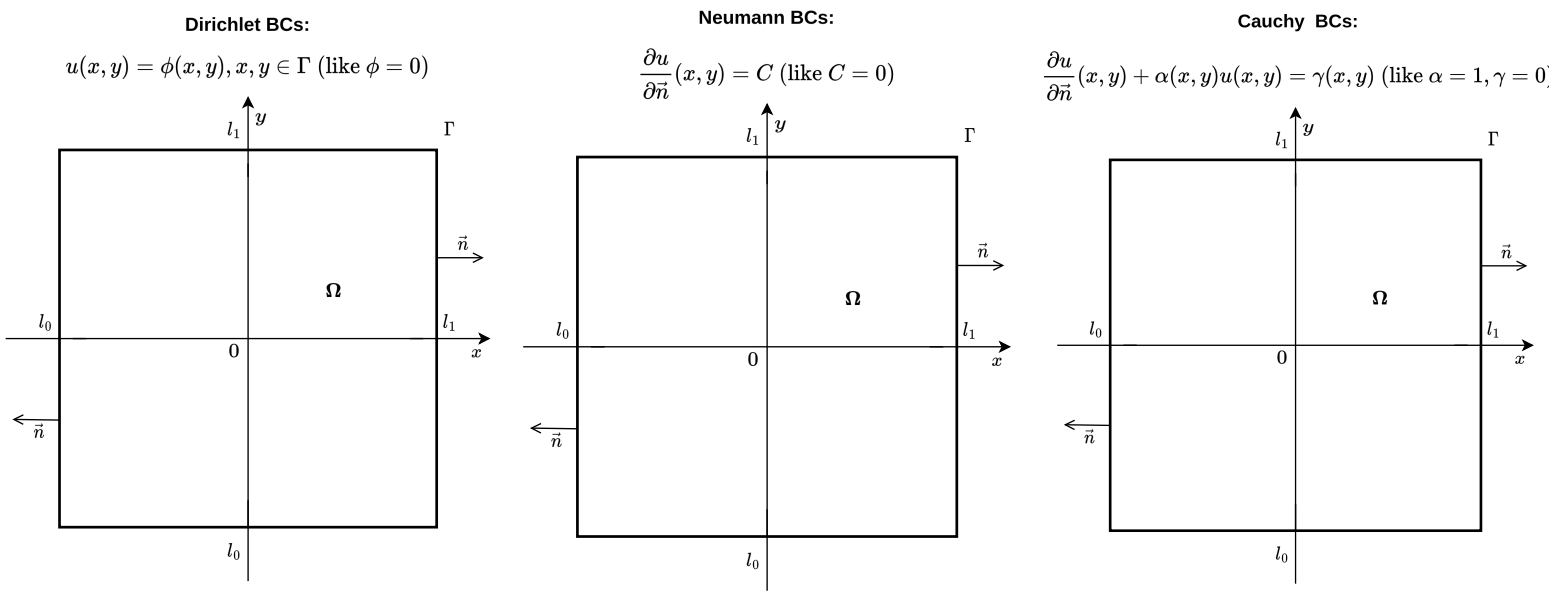
- let $x \in \mathbb{R}^{n^2}$ be the grid points of the 2-dimensional Ω (2D domain Ω);
- $u(x) : \mathbb{R}^{n^d} \to \mathbb{R}$ is the solution to be computed;
- $\rho(x) : \mathbb{R}^{n^2} \to \mathbb{R}$ is a parametric source/forcing term;
- $p(x) : \mathbb{R}^{n^2} \to \mathbb{R}_+$ represents the heterogeneous permeability of the porous medium (non-uniform $p(x)$ varies from point to point);
- $d(x) : \mathbb{R}^{n^2} \to \mathbb{R}_+$ denotes the heterogeneous diffusivity or viscosity field associated with diffusion;
- $\vec{c}(x) : \mathbb{R}^{n^2} \to \mathbb{R}$ represents the heterogeneous velocity or force vector field responsible for convection.

Varying Boundary Conditions (BCs)

For the heterogeneous fluid PDEs described in Equation (1)-(3), we consider three common classical BCs, Dirichlet, Neumann, and Cauchy:

$$\begin{cases} \text{Dirichlet BCs: } u(x) = \phi(x) \text{ (like } \phi(x) = 0 - \text{zero-Dirichlet)}; \\ \text{Neumann BCs: } \dfrac{\partial u}{\partial \vec{n}}(x) = C \text{ (like } C = 0); \\ \text{Cauchy BCs: } \dfrac{\partial u}{\partial \vec{n}}(x) + \alpha(x)u(x) = \gamma(x) \text{ (like } \alpha(x) = 1, \gamma(x) = 0). \end{cases} \tag{4}$$

Visualize Dirichlet, Neumann, and Cauchy BCs on a 2D domain Ω:



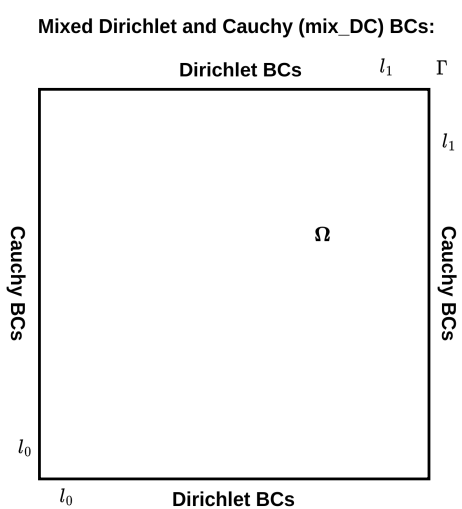
(*a*). Dirichlet BCs (*b*). Neumann BCs (*c*). Cauchy BCs

Varying Boundary Conditions (BCs)

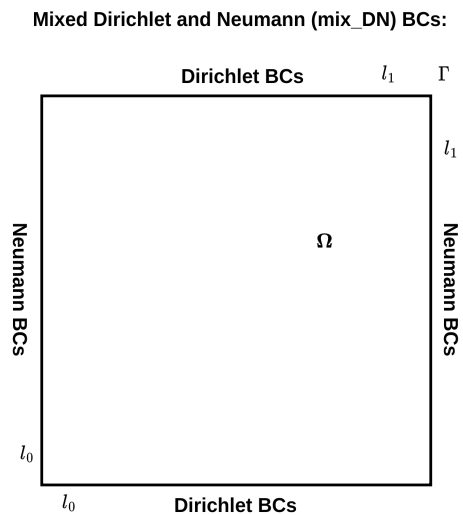
For the heterogeneous fluid PDEs described in Equation (1)-(3), we consider three common classical BCs, Dirichlet, Neumann, and Cauchy:

$$\begin{cases} \text{Dirichlet BCs: } u(x) = \phi(x) \text{ (like } \phi(x) = 0 - \text{zero-Dirichlet)}; \\ \text{Neumann BCs: } \dfrac{\partial u}{\partial \vec{n}}(x) = C \text{ (like } C = 0); \\ \text{Cauchy BCs: } \dfrac{\partial u}{\partial \vec{n}}(x) + \alpha(x)u(x) = \gamma(x) \text{ (like } \alpha(x) = 1, \gamma(x) = 0). \end{cases} \tag{4}$$

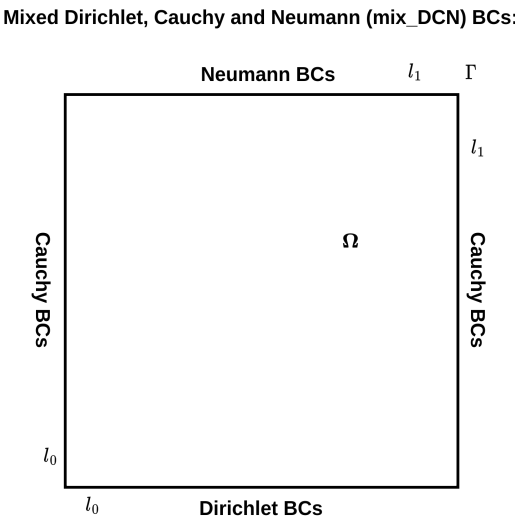
Visualize their mixtures, i.e., the combinations across different boundary:



(*d*). mix_DC BCs



(*e*). mix_DN BCs



(*f*). mix_DCN BCs

Convolution neural networks preconditioning fluid operators

Transfer the differential expression of equations into the discrete form:

$$\text{Fluid PDEs:} \begin{cases} \mathcal{A}([p, \text{or } d, c, \text{BCs}]) = \rho \\ + \text{ the three classical BCs or their three mixtures on 2D domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{FDM, FEM, FVM, } \textbf{FFT}, \ldots \\ A([\text{p}, \text{or d}, \text{c}, \text{BCs}])\text{u} = \text{b}, \text{ with LinearOp } A(*) \in \mathbb{C}^{n \times n}, \ \text{u}, \ \text{b} \in \mathbb{C}^n \end{cases}$$

After discretization, numerical linear algebra methods, like subspace methods, can be used to solve the discrete linear systems. However,

- purely using subspace methods without preconditioning is ineffective;
- generate a properly algebraic preconditioner could be possible (if the n is not too big) but is as challenging as solve the system directly;
- generally, the algebraic preconditioning needs to be re-generated for each of the parametric equations;

* except for the varying heterogeneous parameters, the application of different BCs also strongly effects the property of the linear systems

Convolution neural networks preconditioning fluid operators

Transfer the differential expression of equations into the discrete form:

$$\text{Fluid PDEs:} \begin{cases} \mathcal{A}([p, \text{or } d, c, \text{BCs}]) = \rho \\ + \text{ the three classical BCs or their three mixtures on 2D domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{FDM, FEM, FVM, } \textbf{FFT}, \ldots \\ A([\text{p}, \text{or d}, \text{c}, \text{BCs}])\text{u} = \text{b}, \text{ with LinearOp } A(*) \in \mathbb{C}^{n \times n}, \ \text{u}, \ \text{b} \in \mathbb{C}^n \end{cases}$$

In recent decade, the thrived neural networks (NNs) solvers, like the physics-informed neural networks (PINNs)[a], is used to solve PDEs without discretization. However, these NNs solvers

- could be costly in training[b], and may fail to solve some challenging PDEs if without finely tuning of the hyper-parameters of the NNs;
- solve the PDEs without the theoretical convergence guarantee;
- generally reach limited accuracy and exhibit limited or NO network generalizability, thus re-training is required even it is costly.

[a] Lu et al., Physics-Informed Neural Networks with Hard Constraints for Inverse Design. SISC. 2021

[b] Strubell et al., Energy and policy considerations for deep learning in NLP. ACL meeting, Italy. 2019

Convolution neural networks preconditioning fluid operators

Transfer the differential expression of equations into the discrete form:

Fluid PDEs: $\begin{cases} \mathcal{A}([p, \text{or } d, c, \text{BCs}]) = \rho \\ + \text{ the three classical BCs or their three mixtures on 2D domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{FDM, FEM, FVM, } \textbf{FFT}, \ldots \\ A([\text{p}, \text{or d}, \text{c}, \text{BCs}])\text{u} = \text{b}, \text{ with LinearOp } A(*) \in \mathbb{C}^{n \times n}, \ \text{u}, \ \text{b} \in \mathbb{C}^n \end{cases}$

* **Goal of this work** $\Rightarrow$ To learn neural operator $\mathcal{F}_\theta$ that approximates

$A(*)^{-1}$ by $\mathcal{F}_\theta([b, p, \text{or } d, c, \text{BCs}]) \longrightarrow \text{u}_\theta \sim A([b, p, \text{or } d, c, \text{BCs}])^{-1}\text{b}$

Operator $\mathcal{F}_\theta$ can be used as a **flexible preconditioner** for the FGMRES method to accelerate the fluid equations with varying b, varying p, or varying d, c, varying BCs, and varying domain Ω **without re-training**.

* **Loss-function:**

$$\min_\theta \frac{\|A\text{u}_\theta - \boldsymbol{b}\|_2^2}{\|\boldsymbol{b}\|_2^2} \tag{6}$$

The generalized Arnoldi relation

Originally, the FGMRES method[a] have the generalized Arnoldi relation

$$AZ_m = V_{m+1}\bar{H}_m, \tag{7}$$

where $V_{m+1} = [v_1, ..., v_{m+1}]$ is the Krylov basis, and $Z_m = [z_1, ..., z_m]$ is the preconditioned Krylov basis. $z_j = M_j v_j \ (j = 1, \ldots, m)$ with $M_j \sim A^{-1}$.

In our NNs preconditioned (CNN)-FGMRES case, we have

$$z_j = \mathcal{F}_\theta([v_j, *, *]), \ j = 1, \ldots m, \tag{8}$$

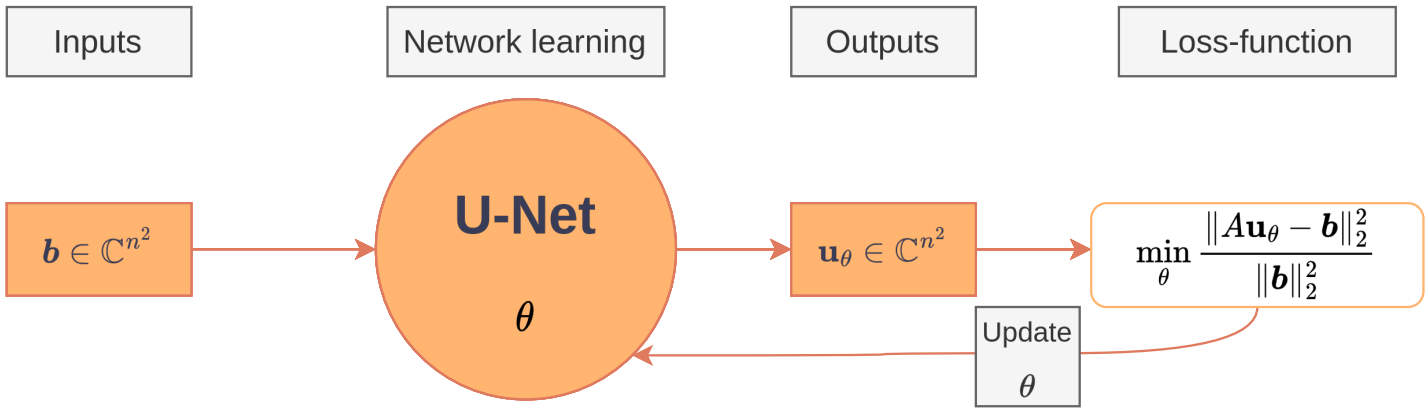
where $\mathcal{F}_\theta$ is the trained non-linear neural operator satisfies $\mathcal{F}_\theta \sim A^{-1}$.

This part is computed with single machine precision (float 32), the one used in the training process, and other parts are in double precision.

$\Rightarrow$ Given there is no information about the data structure of the Krylov basis, the neural operator $\mathcal{F}_\theta$ is trained with randomly generated datasets.

[a] Saad, A Flexible Inner-Outer Preconditioned GMRES Algorithm. SISC. 1993

Neural operator preconditioning for Poisson equations (eqs.)



Training (on 4 V100 GPUs, depth $= 4$, train. time: 27.96 mins):

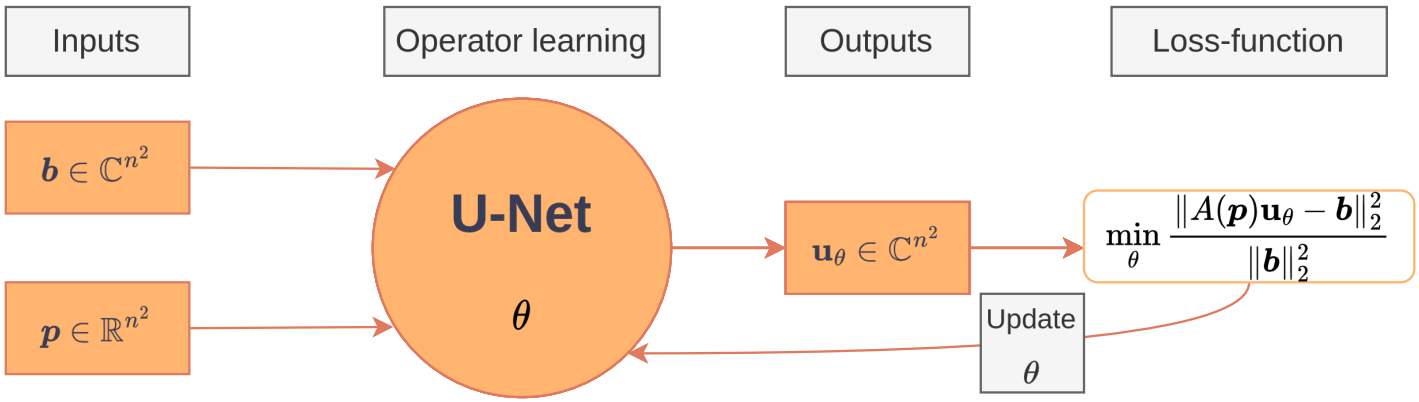
- Fixed grid point $x \in \mathbb{R}^{n^2}$ in the 2-dimensional domain: 64×64 thus $n = 64$
- Random source term $b \in \mathbb{C}^{n^2} \sim \mathcal{N}(0, 1)$ satisfying standard normal distribution
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 831 K
- Set max_epoch=500 with 10,000 training dataset

* Training with zero-Dirichlet BCs is suffice to obtain effective preconditioning operator

Testing:

☺ Trained U-Net preconditioner can accelerate the solution of Poisson Eq. (1) with varying b, BCs, and varying domain size Ω (because of the discretisation invariance property from the convolution property)

Neural operator preconditioning for Darcy flows



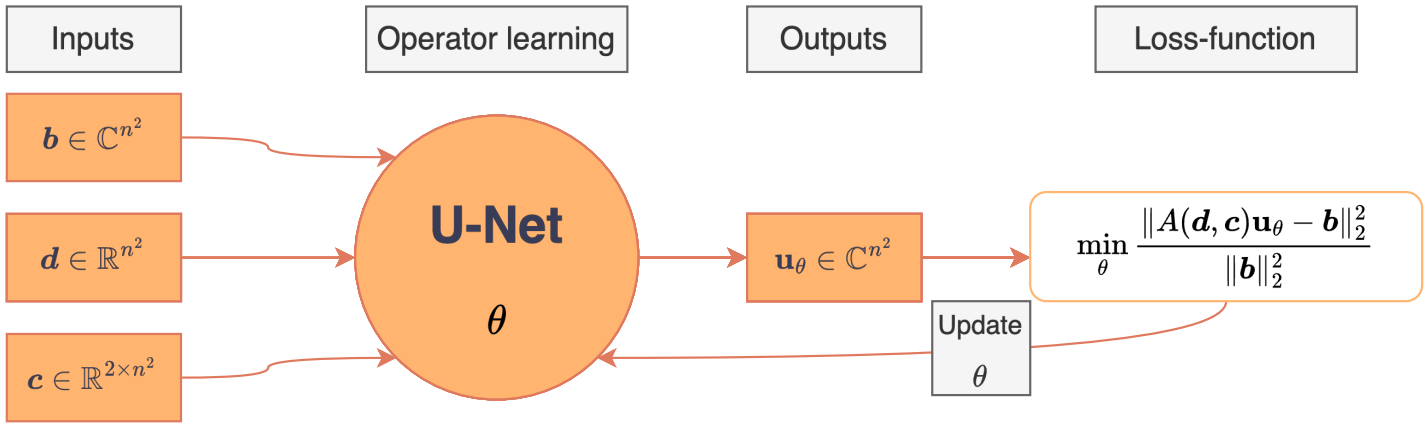
Training (on 4 V100 GPUs, depth $= 4$, train. time: 29.52 mins):

- Fixed grid point $x \in \mathbb{R}^{n^2}$ in the 2-dimensional domain: 64×64 thus $n = 64$
- Random source term $b \in \mathbb{C}^{n^2} \sim \mathcal{N}(0, 1)$ satisfying standard normal distribution
- Random heterogeneous permeability $p \in \mathbb{R}^{n^2+} \sim \mathcal{U}(1, 2)$ uniformly distributed on the interval [1, 2]
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 831 K

Testing:

☺ Trained U-Net preconditioner can accelerate the solution of Darcy flows. (2) with varying b, varying p, varying BCs, and varying domain size Ω

Neural operator preconditioning for Convection-Diffusion eqs.



Training (on 4 V100 GPUs, depth $= 4$, train. time: 38.58 mins):

- Fixed grid point x $\in \mathbb{R}^{n^2}$ in the 2-dimensional domain: 64 $\times$ 64 thus $n = 64$
- Random source term b $\in \mathbb{C}^{n^2} \sim \mathcal{N}(0, 1)$ satisfying standard normal distribution
- Random heterogeneous diffusion term d $\in \mathbb{R}^{n^2+} \sim \mathcal{U}(1, 2)$ uniformly distributed on the interval [1, 2]
- Random heterogeneous convection term c $\in \mathbb{R}^{2\times n^2} \sim \mathcal{U}(1, 2)$
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 832 K

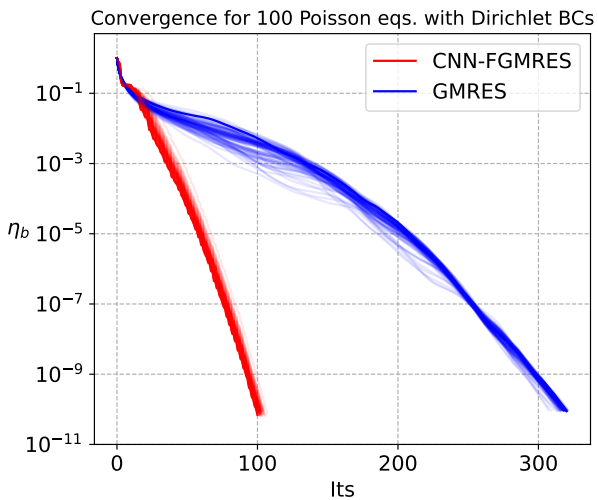
Testing:

☺ Trained U-Net preconditioner can accelerate the solution of Convection-Diffusion equations (3) with varying b, varying d, varying c, varying BCs, and varying domain Ω

General testing results

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

- Stopping criterion: $\eta_b = \frac{\|A\mathbf{u} - \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \varepsilon$ with
  $\varepsilon = 10^{-10}$ for Poisson eqs. (1);
  $\varepsilon = 10^{-8}$ for Darcy flows (2) and Convection-Diffusion eqs. (3);
- Maximum dimension of the Krylov search space is set to be 512;
  (if necessary, apply restart every 512 Krylov step with restart equals 10)
- Source term $\mathrm{b} = x + 0j \in \mathbb{C}^{64^2}$ with $x \in \mathbb{R}^{64^2} \sim \mathcal{N}(0, 1)$.



Poisson eqs.: $A\mathrm{u} = \mathrm{b}$, with
epoch $= 479$, val_loss $= 0.0721$

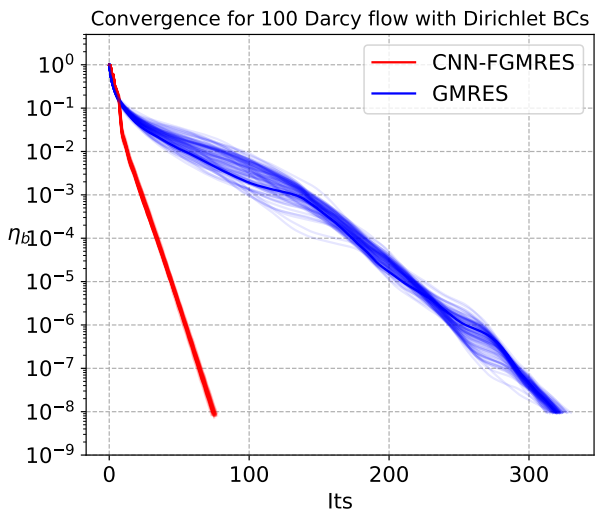
# $(its_{max}, its_{avg}, its_{min})$	# ET
(108, 102, 100)	135.76s
(322, 318, 308)	5616.44s
ET_{training}	27.96 mins
$L_{\text{tt}}^{\text{GMRES}}$	32
$L_{\text{tt}}^{\text{CNN-FGMRES}}$	275

Consumed its & GPU time

General testing results

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

- Stopping criterion: $\eta_b = \frac{\|A\mathbf{u} - \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \varepsilon$ with
  $\varepsilon = 10^{-10}$ for Poisson eqs. (1);
  $\varepsilon = 10^{-8}$ for Darcy flows (2) and Convection-Diffusion eqs. (3);
- Maximum dimension of the Krylov search space is set to be 512;
  (if necessary, apply restart every 512 Krylov step with restart equals 10)
- Source term $\mathrm{b} = x + 0j \in \mathbb{C}^{64^2}$ with $x \in \mathbb{R}^{64^2} \sim \mathcal{N}(0, 1)$.



# $(its_{max}, its_{avg}, its_{min})$	# ET
(78, 75, 74)	71.17s
(331, 321, 313)	5106.56s
ET_{training}	29.52 mins
$L_{\text{tt}}^{\text{GMRES}}$	34
$L_{\text{tt}}^{\text{CNN-FGMRES}}$	314

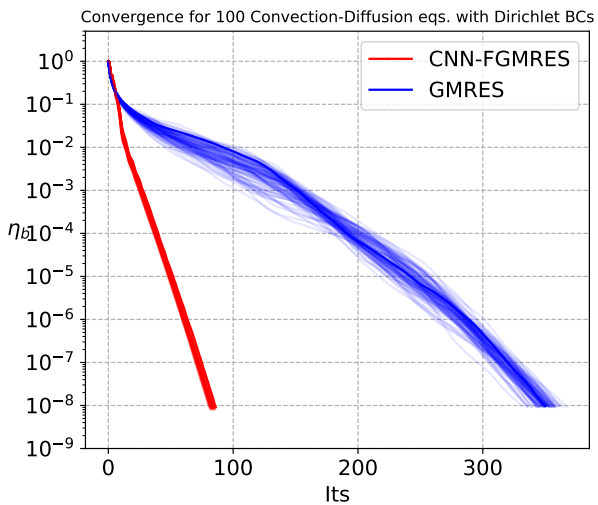
Consumed its & GPU time

Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$, with
epoch $= 470$, val_loss $= 0.2472$

General testing results

Solve three different fluid equations (eqs.): $A([\mathrm{p, or\ d, c, BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

- Stopping criterion: $\eta_b = \frac{\|A\mathbf{u} - \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \varepsilon$ with
  $\varepsilon = 10^{-10}$ for Poisson eqs. (1);
  $\varepsilon = 10^{-8}$ for Darcy flows (2) and Convection-Diffusion eqs. (3);
- Maximum dimension of the Krylov search space is set to be 512;
  (if necessary, apply restart every 512 Krylov step with restart equals 10)
- Source term $\mathrm{b} = x + 0j \in \mathbb{C}^{64^2}$ with $x \in \mathbb{R}^{64^2} \sim \mathcal{N}(0, 1)$.



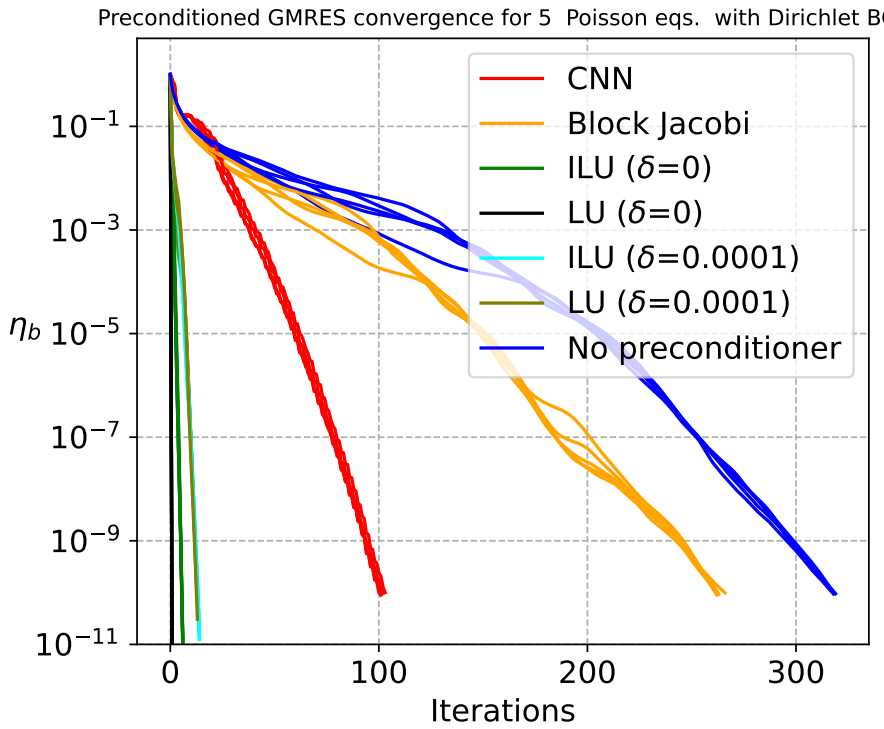
Conv-Diff: $A([\mathrm{d, c}])\mathrm{u} = \mathrm{b}$, with
epoch $= 491$, val_loss $= 0.3383$

# $(its_{max}, its_{avg}, its_{min})$	# ET
(87, 84, 81)	93.52s
(368, 351, 332)	6023.95s
$ET_{\mathbf{training}}$	38.58 mins
$L_{\mathbf{tt}}^{\mathbf{GMRES}}$	34
$L_{\mathbf{tt}}^{\mathbf{CNN\text{-}FGMRES}}$	398

Consumed its & GPU time

Trained network preconditioner VS Algebraic preconditioner

Solve $A(c)u = b$ on 2D domain 64×64 (i.e., $A(c) \in \mathbb{C}^{64^2 \times 64^2}$):



5 preconditioned GMRES results for the Poisson equations (eqs.)

# $(its_{max}, its_{avg}, its_{min})$	# *ET*
CNN: (104, 102, 102)	8.34s
BJ: (267, 264, 263)	22.07s
ILU, $\delta = 0$: (7, 7, 7)	0.14s
LU, $\delta = 0$: (2, 2, 2)	0.12s
ILU: (15, 15, 15)	0.35s
LU: (14, 14, 14)	0.84s
No Pre: (320, 319, 319)	40.29s

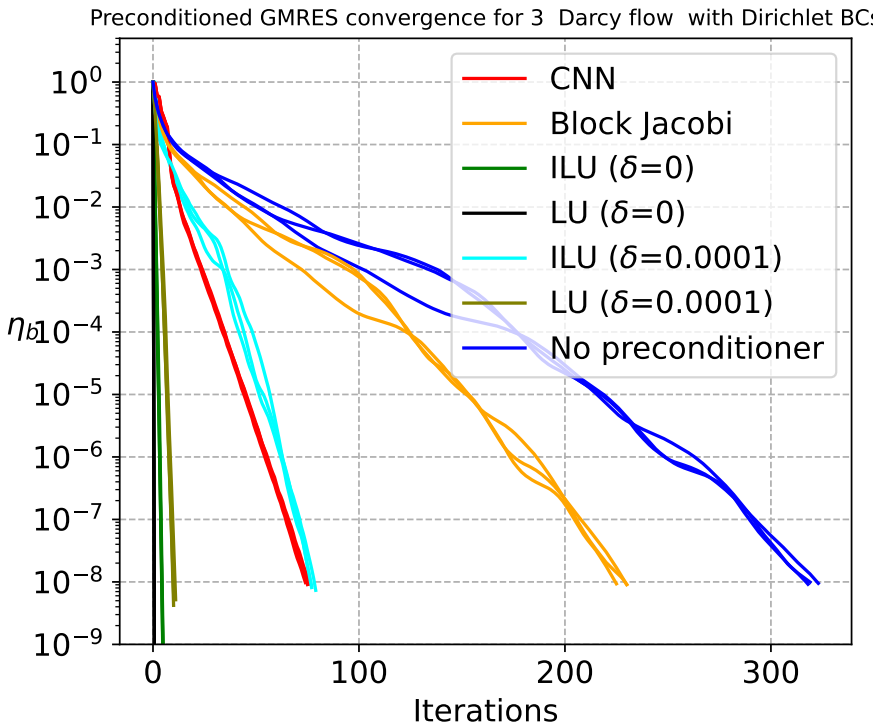
Consumed its & CPU time

- Effectiveness of trained network preconditioner compared to `sp(i)lu`$(\tilde{A})$ with varying δ.

$$\begin{cases} \tilde{a}(c)_{ii} = a(c)_{ii}, \\ \tilde{a}(c)_{ij} = a(c)_{ij}, & \text{if } |a(c)_{ij}| > \delta |a(c)_{ii}|, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and define } \eta = \frac{\|\tilde{A} - A\|_F}{\|A\|_F}.$$

Trained network preconditioner VS Algebraic preconditioner

Solve $A(c)u = b$ on 2D domain 64×64 (i.e., $A(c) \in \mathbb{C}^{64^2 \times 64^2}$):



3 preconditioned GMRES results for Darcy flow

# $(its_{max}, its_{avg}, its_{min})$	# ET
CNN: (77, 76, 75)	3.58s
BJ: (231, 229, 226)	8.05s
ILU, $\delta = 0$: (6, 6, 6)	0.09s
LU, $\delta = 0$: (2, 2, 2)	0.02s
ILU: (80, 78, 78)	1.41s
LU: (12, 11, 11)	0.32s
No Pre: (324, 321, 319)	20.59s

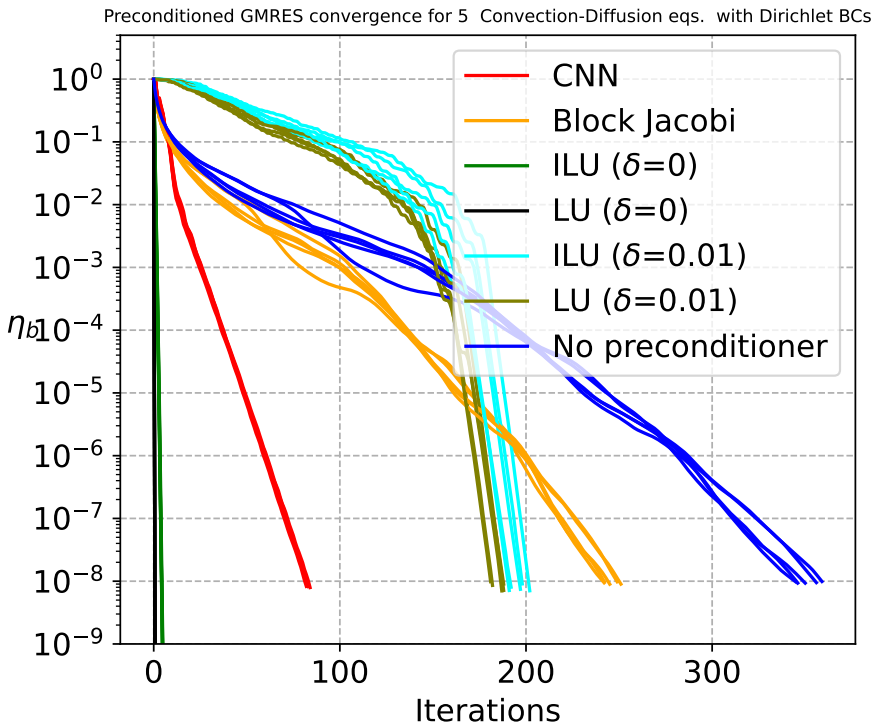
Consumed its & CPU time

- Effectiveness of trained network preconditioner compared to `sp(i)lu`$(\tilde{A})$ with varying δ.

$$\begin{cases} \tilde{a}(c)_{ii} = a(c)_{ii}, \\ \tilde{a}(c)_{ij} = a(c)_{ij}, \quad \text{if } |a(c)_{ij}| > \delta |a(c)_{ii}|, \\ 0 \qquad\qquad\quad \text{otherwise}, \end{cases} \quad \text{and define } \eta = \frac{\|\tilde{A} - A\|_F}{\|A\|_F}.$$

Trained network preconditioner VS Algebraic preconditioner

Solve $A(c)u = b$ on 2D domain 64×64 (i.e., $A(c) \in \mathbb{C}^{64^2 \times 64^2}$):



5 preconditioned GMRES results for the Convection–Diffusion equations (eqs.)

# $(its_{max}, its_{avg}, its_{min})$	# *ET*
CNN: (85, 83, 83)	5.47s
BJ: (252, 246, 243)	16.67s
ILU, $\delta = 0$: (6, 6, 6)	0.19s
LU, $\delta = 0$: (2, 2, 2)	0.10s
ILU: (203, 197, 192)	9.84s
LU: (189, 186, 182)	7.80s
No Pre: (360, 352, 346)	47.64s

Consumed its & CPU time

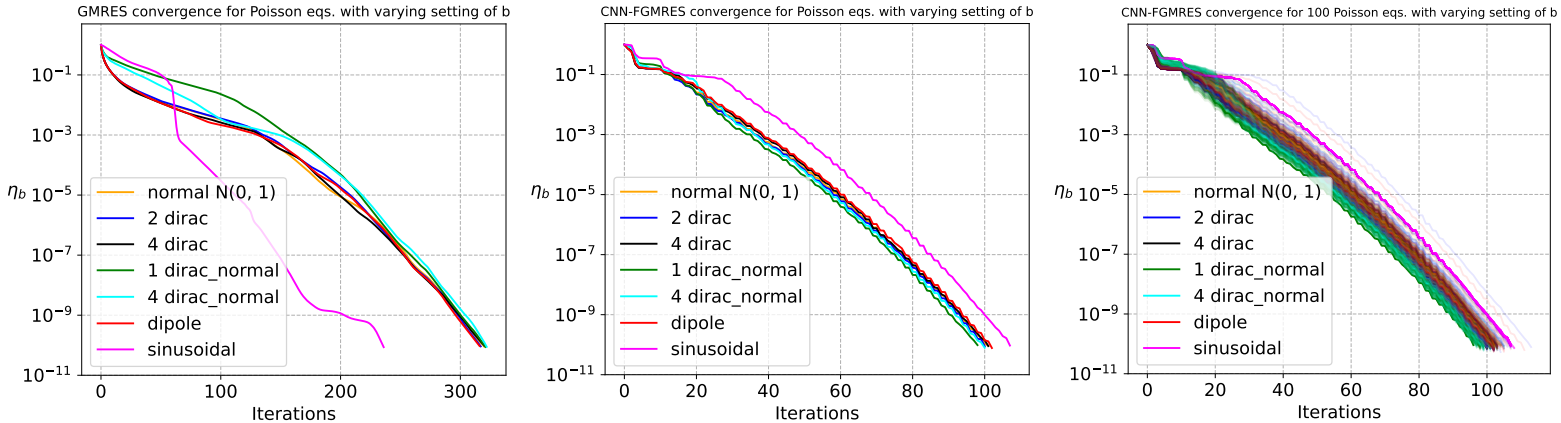
- Effectiveness of trained network preconditioner compared to `sp(i)lu`$(\tilde{A})$ with varying δ.

$$\begin{cases} \tilde{a}(c)_{ii} = a(c)_{ii}, \\ \tilde{a}(c)_{ij} = a(c)_{ij}, \\ 0 \end{cases} \quad \begin{array}{l} \\ \text{if } |a(c)_{ij}| > \delta |a(c)_{ii}|, \\ \text{otherwise,} \end{array} \quad \text{and define } \eta = \frac{\|\tilde{A} - A\|_F}{\|A\|_F}.$$

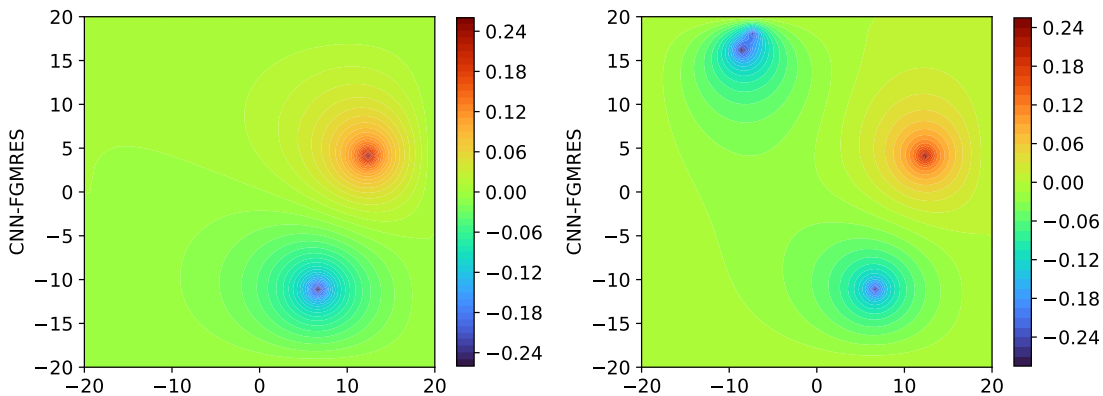
1. Varying the source term b

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

Results of parametric Poisson eqs.: $A\mathrm{u} = \mathrm{b}$



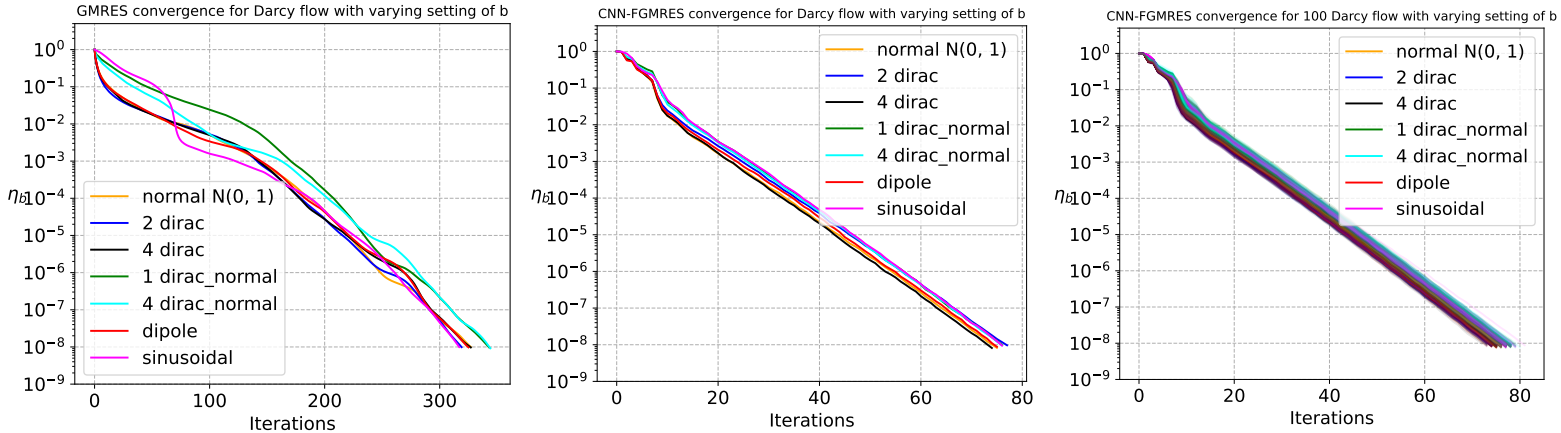
(*a*). 1 GMRES result (*b*). 1 CNN-FGMRES result (*c*). 100 CNN-FGMRES results



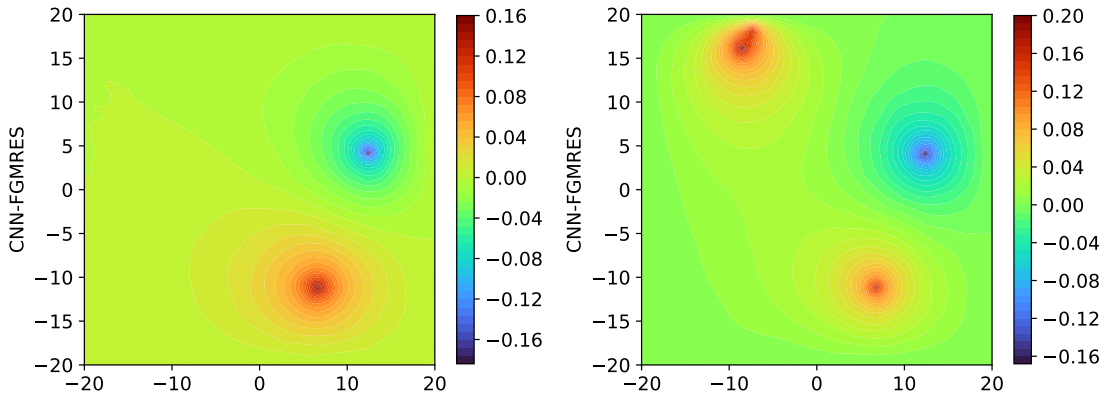
1. Varying the source term b

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

Results of heterogeneous Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$



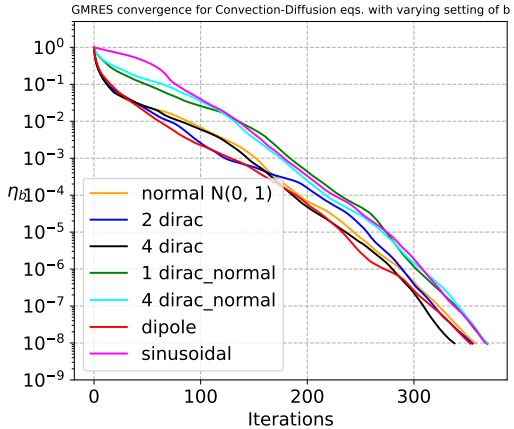
(*a*). 1 GMRES result (*b*). 1 CNN-FGMRES result (*c*). 100 CNN-FGMRES results

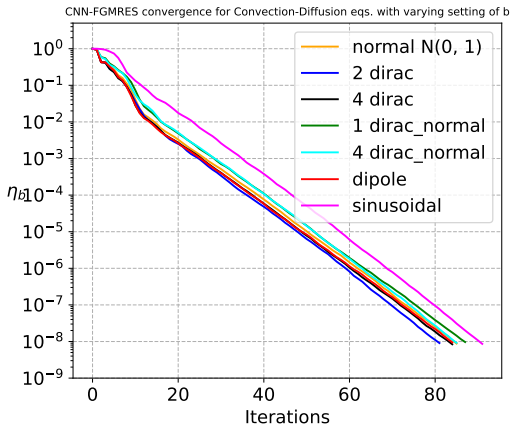


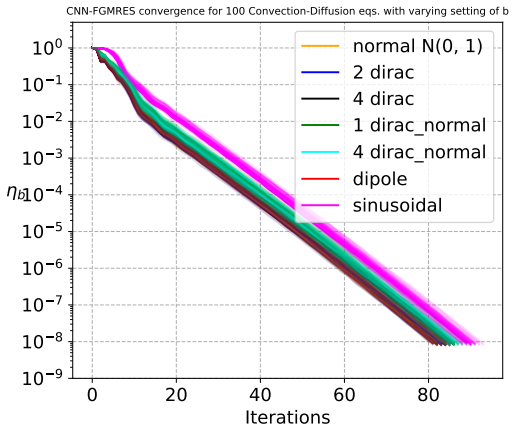
1. Varying the source term b

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with zero-Dirichlet BCs:

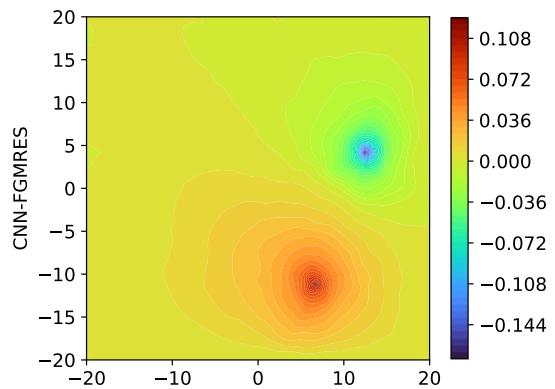
Results of heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}])\mathrm{u} = \mathrm{b}$

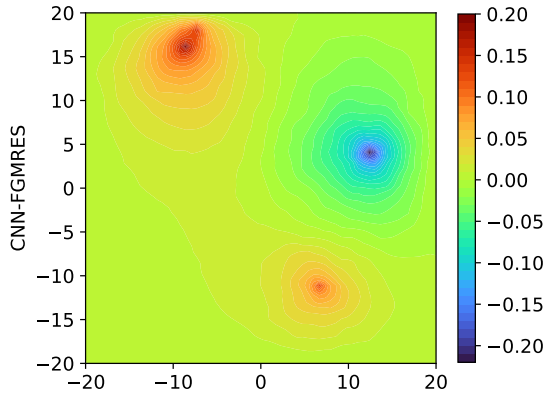






(*a*). 1 GMRES result (*b*). 1 CNN-FGMRES result (*c*). 100 CNN-FGMRES results

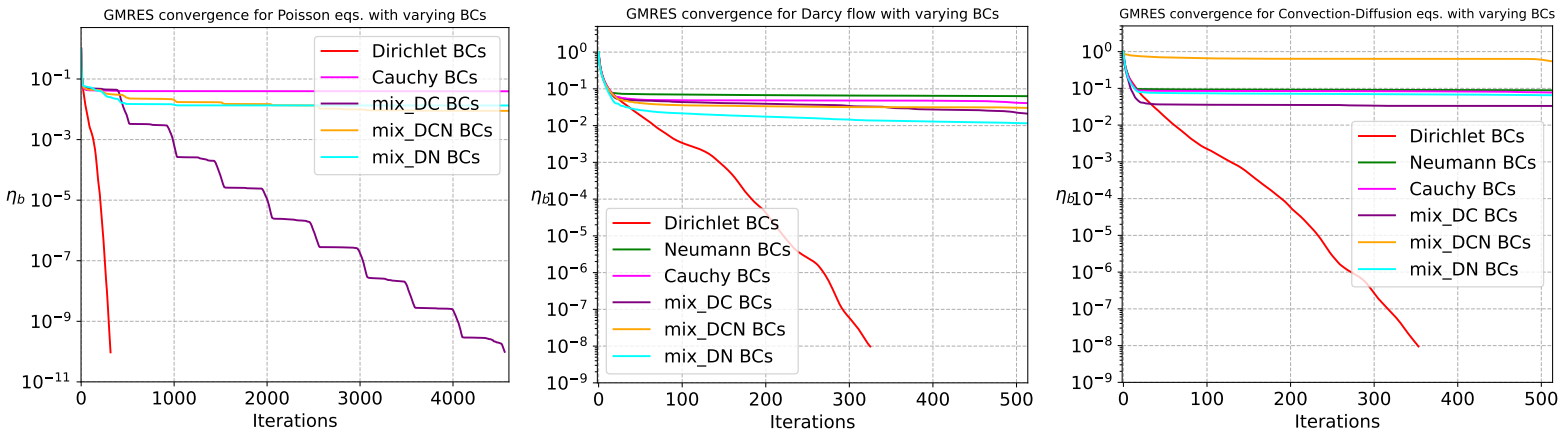




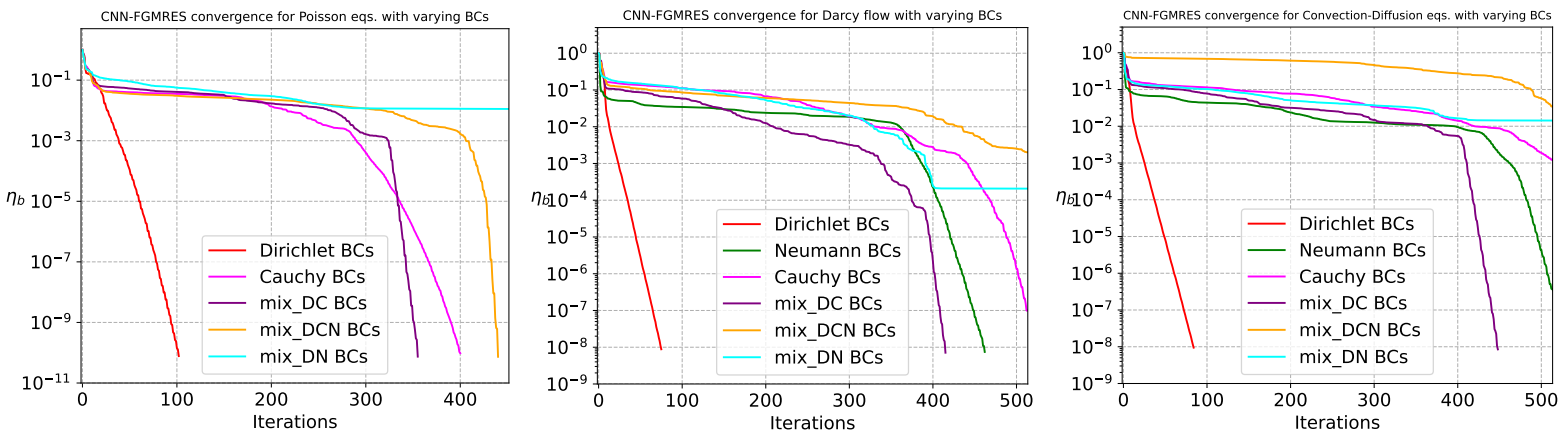
2. Varying the boundary conditions (BCs)

Solve three different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with `dipole` b:

(a). GMRES results



(b). CNN-FGMRES results

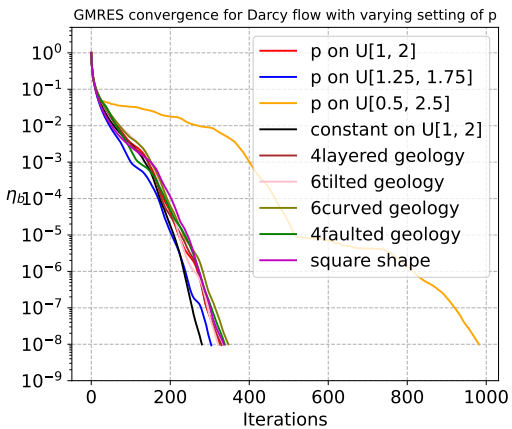


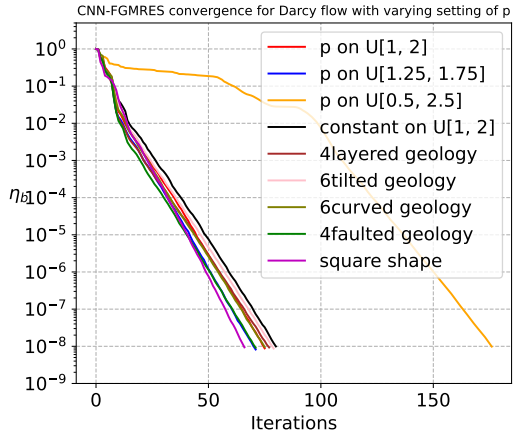
Poisson eqs. | Darcy flows | Convection-Diffusion eqs.

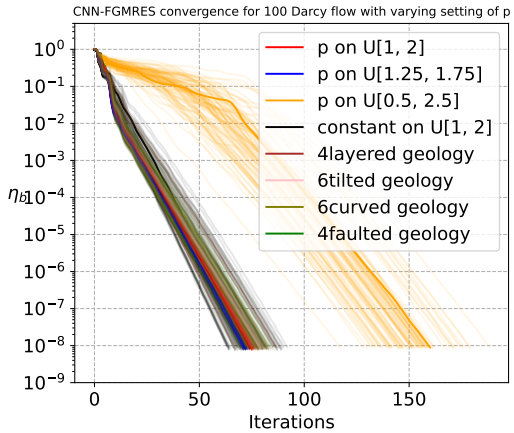
3. Varying the permeability or diffusion term

Solve two different fluid equations (eqs.): $A([\mathrm{p}, \text{or } \mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with `dipole` b:

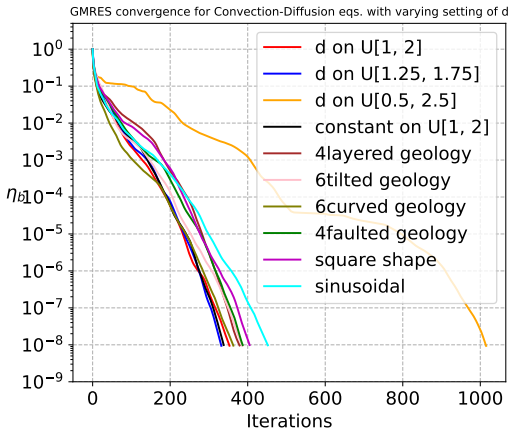
Heterogeneous Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$

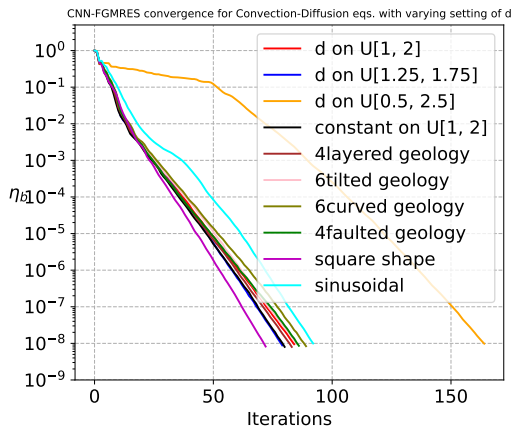


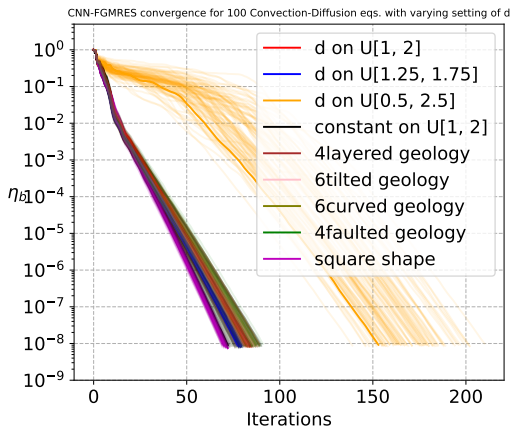




Heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}])\mathrm{u} = \mathrm{b}$

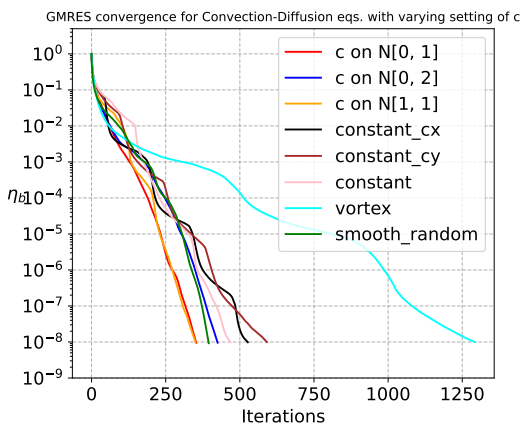


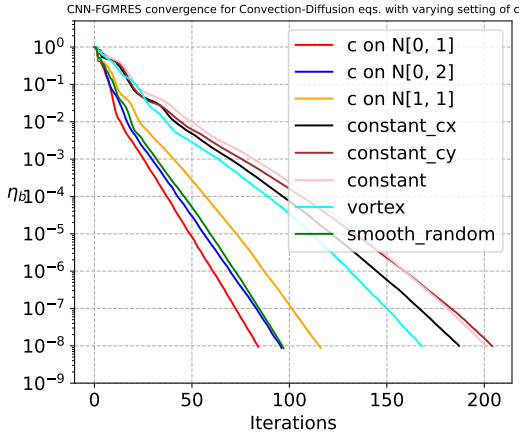


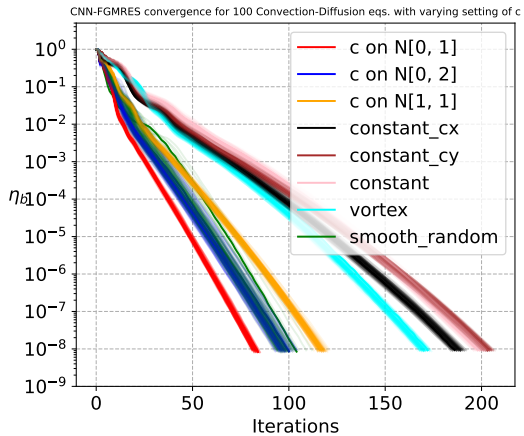


4. Varying the convection term

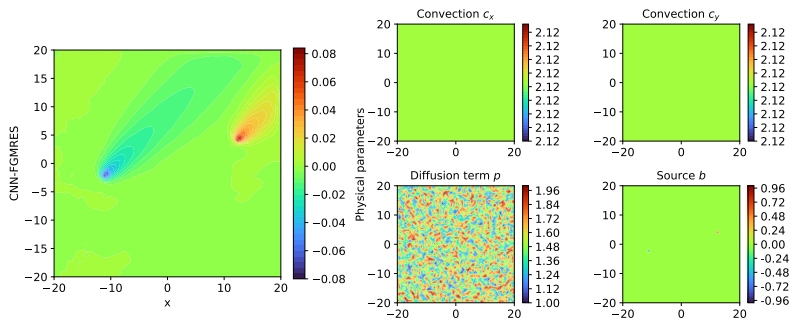
Solve heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with `dipole` b:



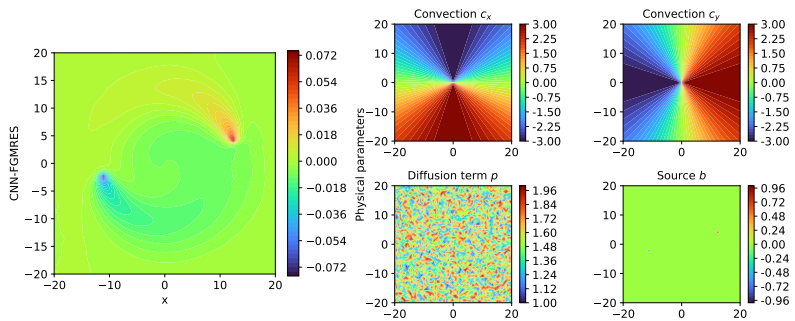




(*a*). 1 GMRES result (*b*). 1 CNN-FGMRES result (*c*). 100 CNN-FGMRES results



(*d*). constant c



(*e*). varying vortex c

4. Varying the convection term

Solve heterogeneous Convection-Diffusion eqs.: $A([\mathsf{d}, \mathsf{c}, \mathrm{BCs}])\mathsf{u} = \mathsf{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with `dipole` b:

Table 12: Numerical results in terms of its, $(its_{max}, its_{avg}, its_{min})$ and ET of GMRES and CNN-FGMRES on a sequence of linear systems ($L = 1, 5$) from the discrete Convection-Diffusion equation (2.11) with varying relative strength $\mathbf{c} \sim$ `constant_cx` to $\mathbf{d} \sim \mathcal{U}(1, 2)$ (for Section 4.3.4).

# $\mathbf{c} \in \mathbb{R}^{2\times 64^2}$ on	Method	L	its or $(its_{max}, its_{avg}, its_{min})$	ET
`constant_cx` with $\alpha_{\mathbf{c}} = 0.5$	GMRES	1	335	58.89s
	CNN-FGMRES	1	91	3.09s
	CNN-FGMRES	5	(94, 92, 92)	9.19s
`constant_cx` with $\alpha_{\mathbf{c}} = 1.5$	GMRES	1	414	77.86s
	CNN-FGMRES	1	132	5.03s
	CNN-FGMRES	5	(133, 131, 130)	15.39s
`constant_cx` with $\alpha_{\mathbf{c}} = 5$	GMRES	1	765	165.68s
	CNN-FGMRES	1	285	55.94s
	CNN-FGMRES	5	(288, 286, 285)	230.39s

Three cases for the Convection-Diffusion eqs.:

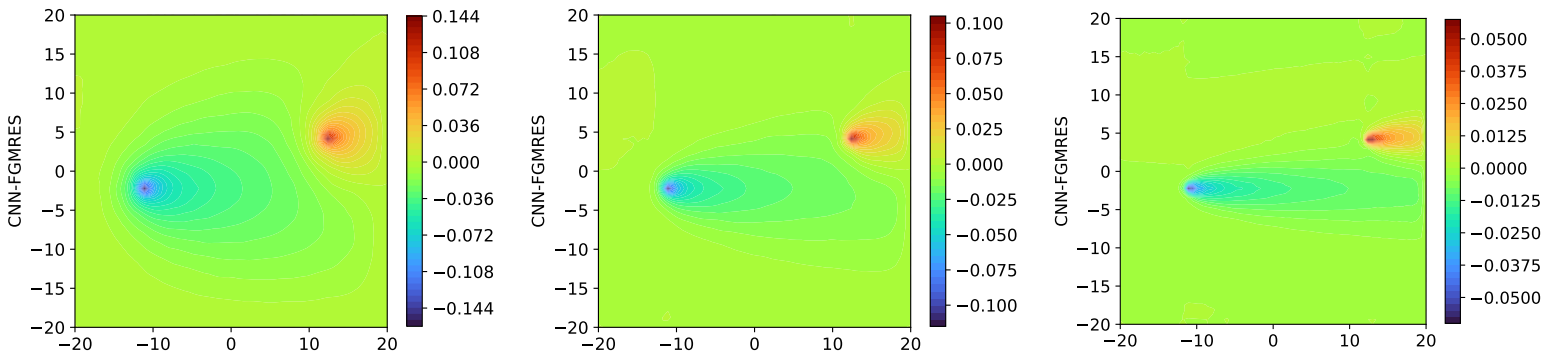
1. **Diffusion-dominant case** with $\frac{\mathbf{d}}{\mathbf{c}} > 1$: diffusion significantly outweighs convection
2. **Balanced case** with $\frac{\mathbf{d}}{\mathbf{c}} \approx 1$: convection and diffusion effects are balanced
3. ***Convection-dominant case** with $\frac{\mathbf{d}}{\mathbf{c}} < 1$: convection dominates diffusion

4. Varying the convection term

Solve heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}, \mathrm{BCs}])\mathrm{u} = \mathrm{b}$ on 2D domain 64×64 (thus $A \in \mathbb{C}^{64^2 \times 64^2}$) with `dipole` b:

Table 12: Numerical results in terms of *its*, $(its_{max}, its_{avg}, its_{min})$ and *ET* of GMRES and CNN-FGMRES on a sequence of linear systems ($L = 1, 5$) from the discrete Convection-Diffusion equation (2.11) with varying relative strength $\mathbf{c} \sim$ `constant_cx` to $\mathbf{d} \sim \mathcal{U}(1, 2)$ (for Section 4.3.4).

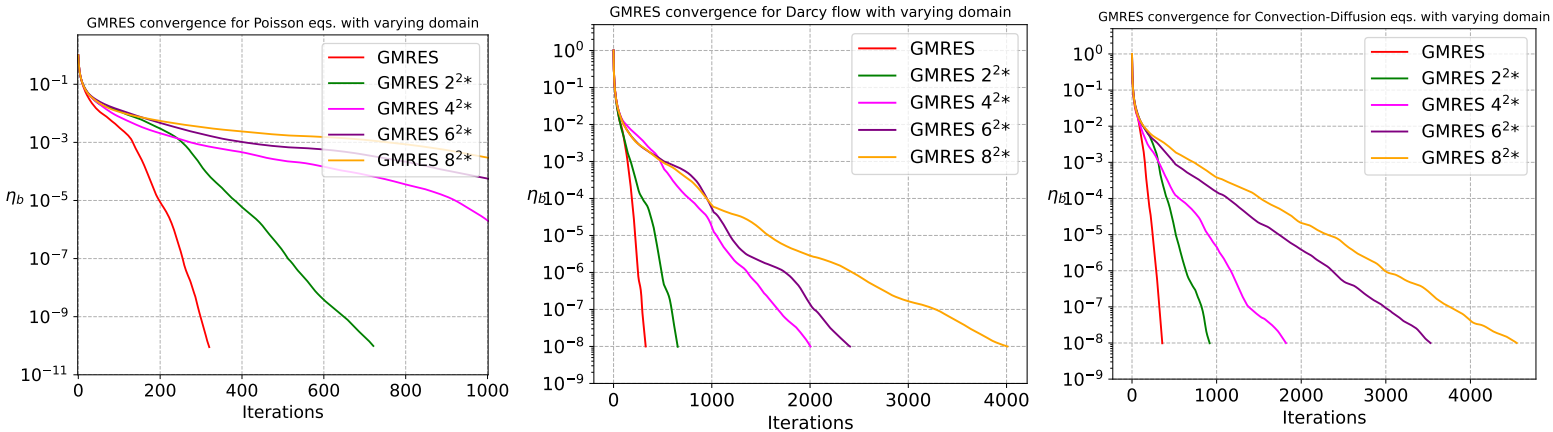
# $\mathbf{c} \in \mathbb{R}^{2\times 64^2}$ on	Method	L	*its* or $(its_{max}, its_{avg}, its_{min})$	*ET*
`constant_cx` with $\alpha_{\mathbf{c}} = 0.5$	GMRES	1	335	58.89s
	CNN-FGMRES	1	91	3.09s
	CNN-FGMRES	5	(94, 92, 92)	9.19s
`constant_cx` with $\alpha_{\mathbf{c}} = 1.5$	GMRES	1	414	77.86s
	CNN-FGMRES	1	132	5.03s
	CNN-FGMRES	5	(133, 131, 130)	15.39s
`constant_cx` with $\alpha_{\mathbf{c}} = 5$	GMRES	1	765	165.68s
	CNN-FGMRES	1	285	55.94s
	CNN-FGMRES	5	(288, 286, 285)	230.39s



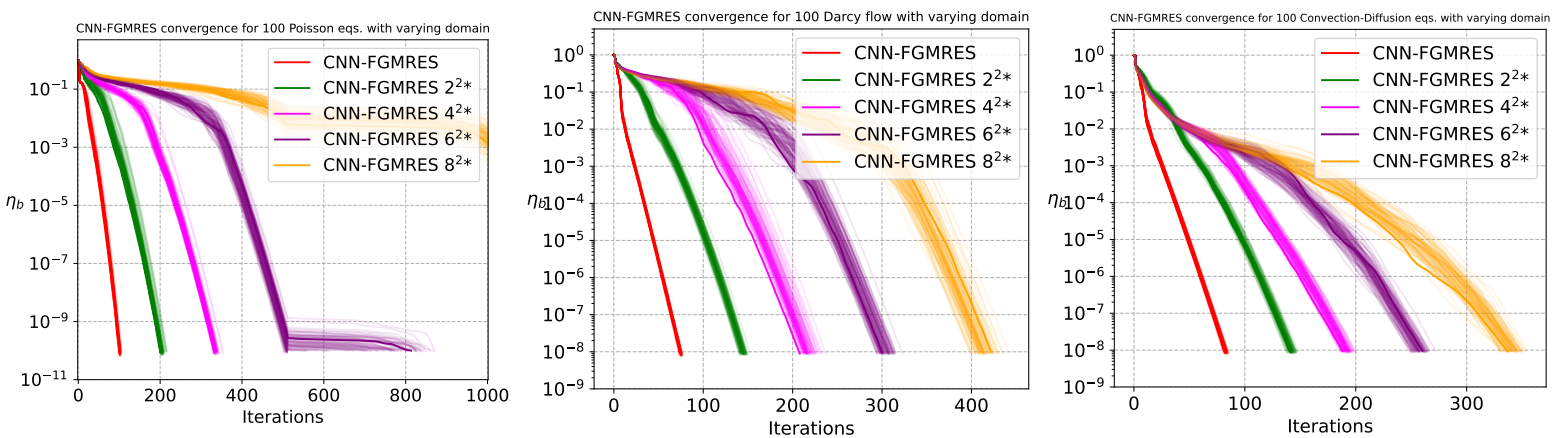
5. Varying the domain size

Solve three different fluid equations (eqs.): $A([\text{p, or d, c, BCs}])\text{u} = \text{b}$ from 2D domain 64×64 until 512×512 (i.e., from linear operator $A \in \mathbb{C}^{64^2 \times 64^2}$ - (F)GMRES to $A \in \mathbb{C}^{512^2 \times 512^2}$ - (F)GMRES 8^2*):

(a). GMRES results



(b). 100 CNN-FGMRES results



Poisson eqs. Darcy flow Convection-Diffusion eqs.

Take home message of this work

Goal: To learn the neural operator preconditioning $\mathcal{F}_\theta$ for accelerating the solution of some parameteric PDEs.

Operator learning + FGMRES method through **flexible preconditioning**

- Convolution neural networks with **U-Net** architecture;
- **Unsupervised training** with randomly generated dataset.

$\Rightarrow$ Trained neural operator preconditioners can be applied to:

- parameteric PDEs with varying heterogeneous parameters, varying boundary conditions, and varying domain size.

Under 2nd Review & Research report version is accessible online at Inria-HAL (https://inria.hal.science/hal-04886933v2).

It can aslo applied to the parameteric Helmholtz equation $\left[\nabla^2 + \left(\frac{\omega}{c(x)}\right)^2\right] u(x) = \rho(x)$ and the flexible FOM method (FFOM)

$\longrightarrow$ **Yanfei Xiang (https://irma.math.unistra.fr/∼xiang/)**

My co-workers from Inria and Cerfacs

- Inria: Luc Giraud and Maksym Shpakovych
- Cerfacs: Paul Mycek and Carola Kruse


Luc Giraud


Paul Mycek


Maksym Shpakovych


Carola Kruse

*We are **Concace** joint Inria team with Airbus CR&T and Cerfacs, France.*

IRMA PEPR PDE-AI Université de Strasbourg

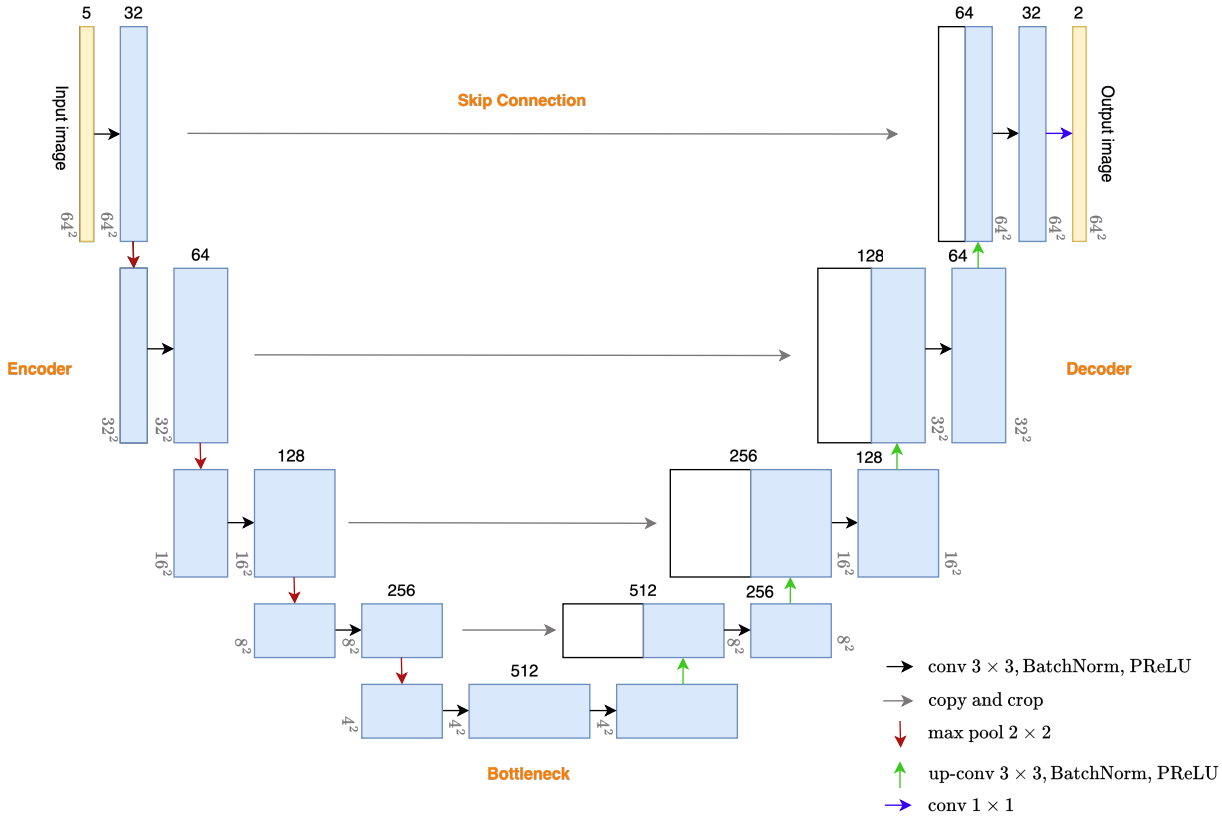
Thank you for your attention !

Questions ?

Appendix

U-Net architecture

U-Net[a] architecture with 4 **depth**:



Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The *x*-*y*-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

[a] Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015

Overview of SciML + numerical iterative methods

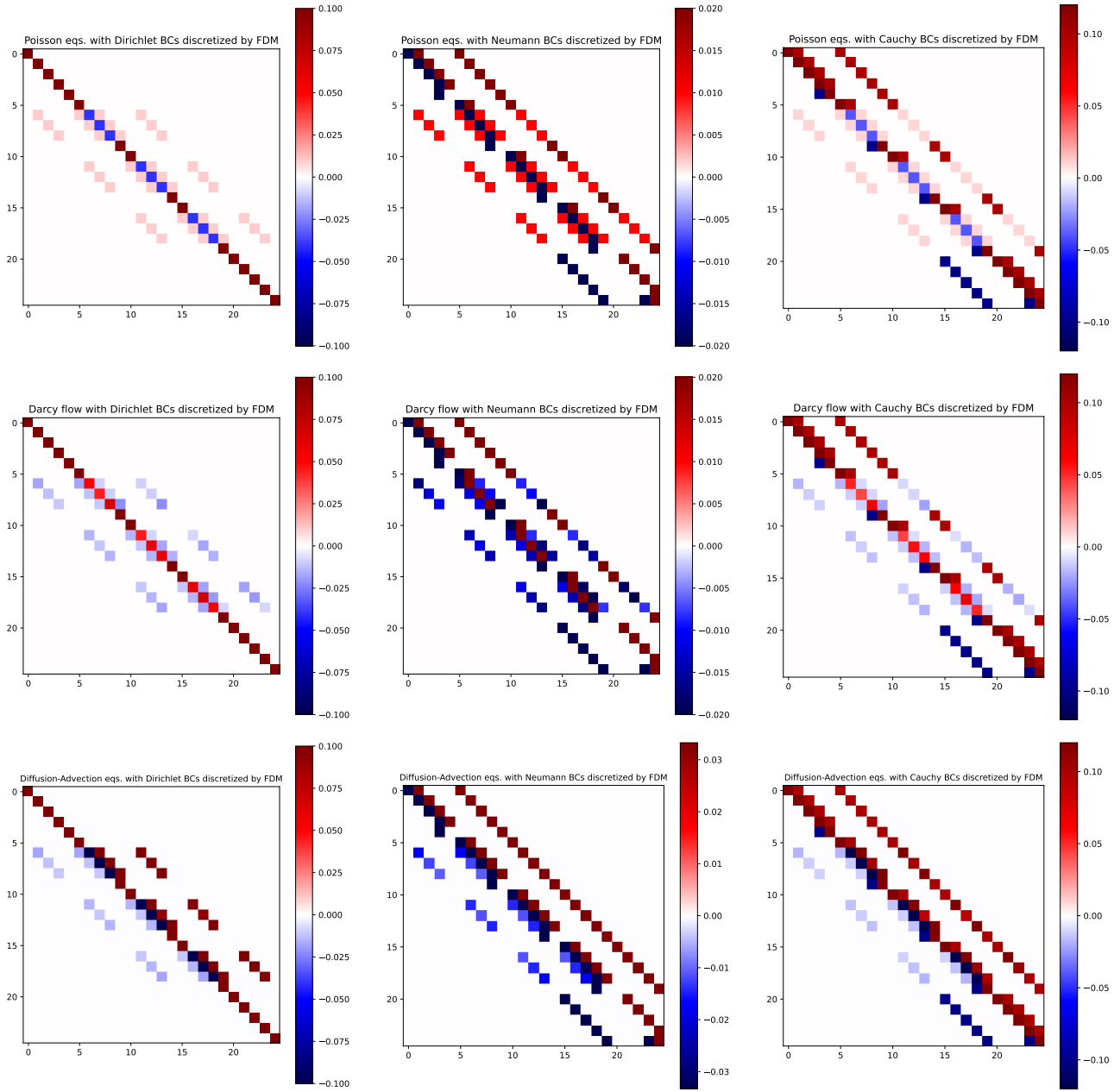
Previous efforts in integrating recent Scientific Machine Learning (SciML) techniques with traditional numerical iterative methods (not exhaustively):

- ***Preconditioning learning***. Kopaničáková & Karniadakis, 2025, Fahy et al., 2024, Li et al., 2024, Azulay & Treister, 2022, Xiang 2022, Ackmann et al., 2021, ...
- *Initial guess learning.* Aghili et al., 2025, Luna et al., 2021, ...
- *Optimal parameters learning.* Khodak et al., 2024, ...
- *Alternative algorithms.* Illarramendi et al., 2020, Rizzuti et al., 2019, ...
- *Multigrid and algebraic multigrid.* Caldana et al., 2024, Dong et al., 2024, Han et al., 2024, Antonietti et al., 2023, Luz et al., 2020, Greenfeld et al., 2019, He & Xu, 2019, Hsieh et al., 2019, ...
- *Domain decomposition methods.* Dolean et al., 2024, Heinlein et al., 2024, Howard et al., 2024, Klawonn et al., 2024, SISC, Klawonn et al., 2024, CSE, Verburg et al., 2024, Moseley et al., 2023, Heinlein et al., 2021, Heinlein et al., 2019, ...
- *Recommendation system.* Chen et al., 2019, Sood, 2019, Yamada et al., 2018, Peairs & Chen, 2011, ...

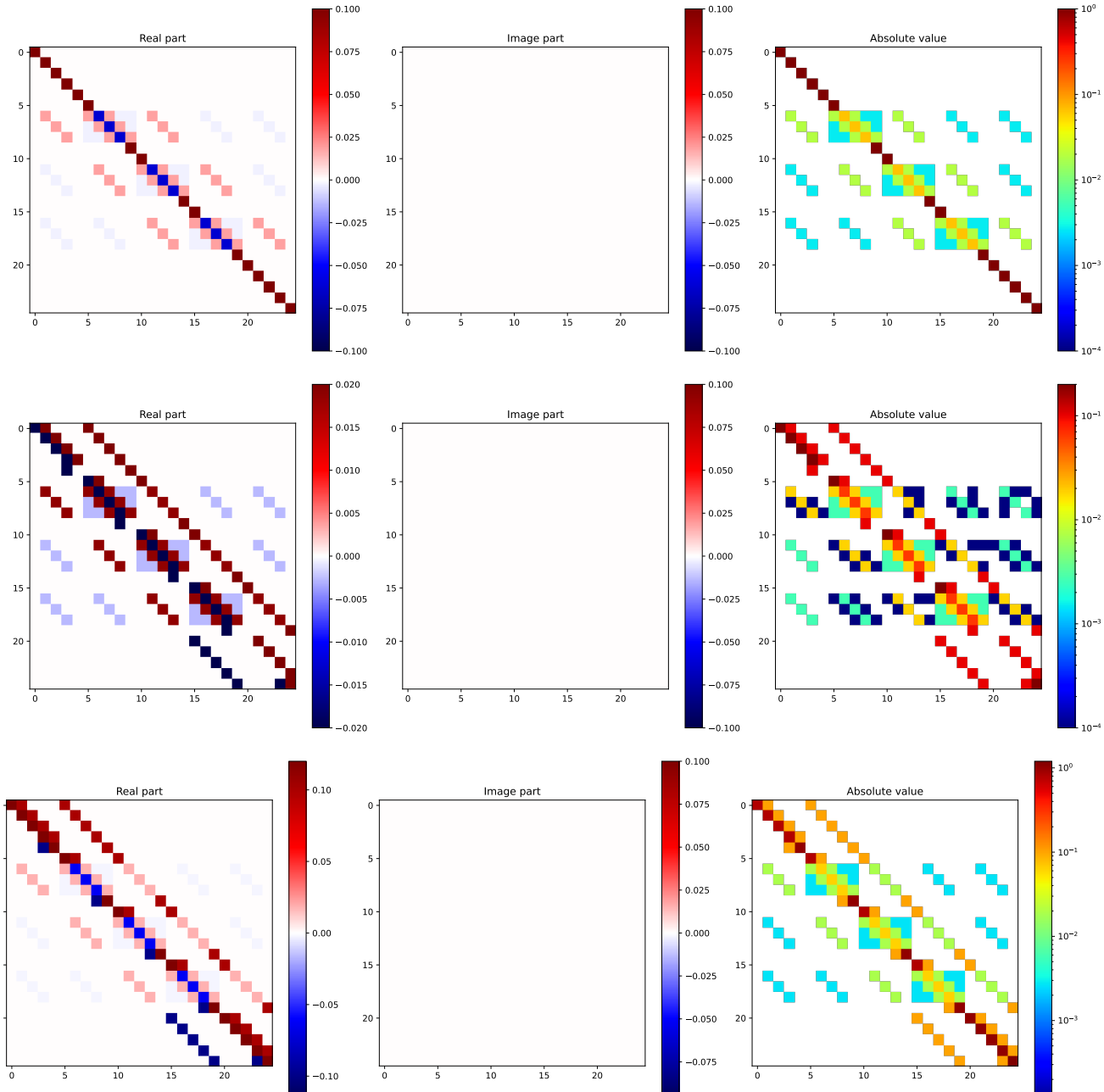
On the other hand, integration SciML with some statistical approaches or non-iterative standard numerical linear algebra methods (not exhaustively):

- *Random feature & Randomization.* Nelsen & Stuart, 2024, Lanthaler & Nelsen, 2023, Chen et al., 2022, ... & Boullé & Townsend, 2023, Boullé et al., 2023, ...
- *Reduced order models.* Borcea et al., 2024, Coscia et al., 2024, Gowrachari et al., 2024, Ivagnes et al., 2024, Quaini et al., 2024, Demo et al., 2023, Gonnella et al., 2023, Ivagnes et al., 2023, Khamlich et al., 2023, Romor et al., 2023, Siena et al., 2023, ...

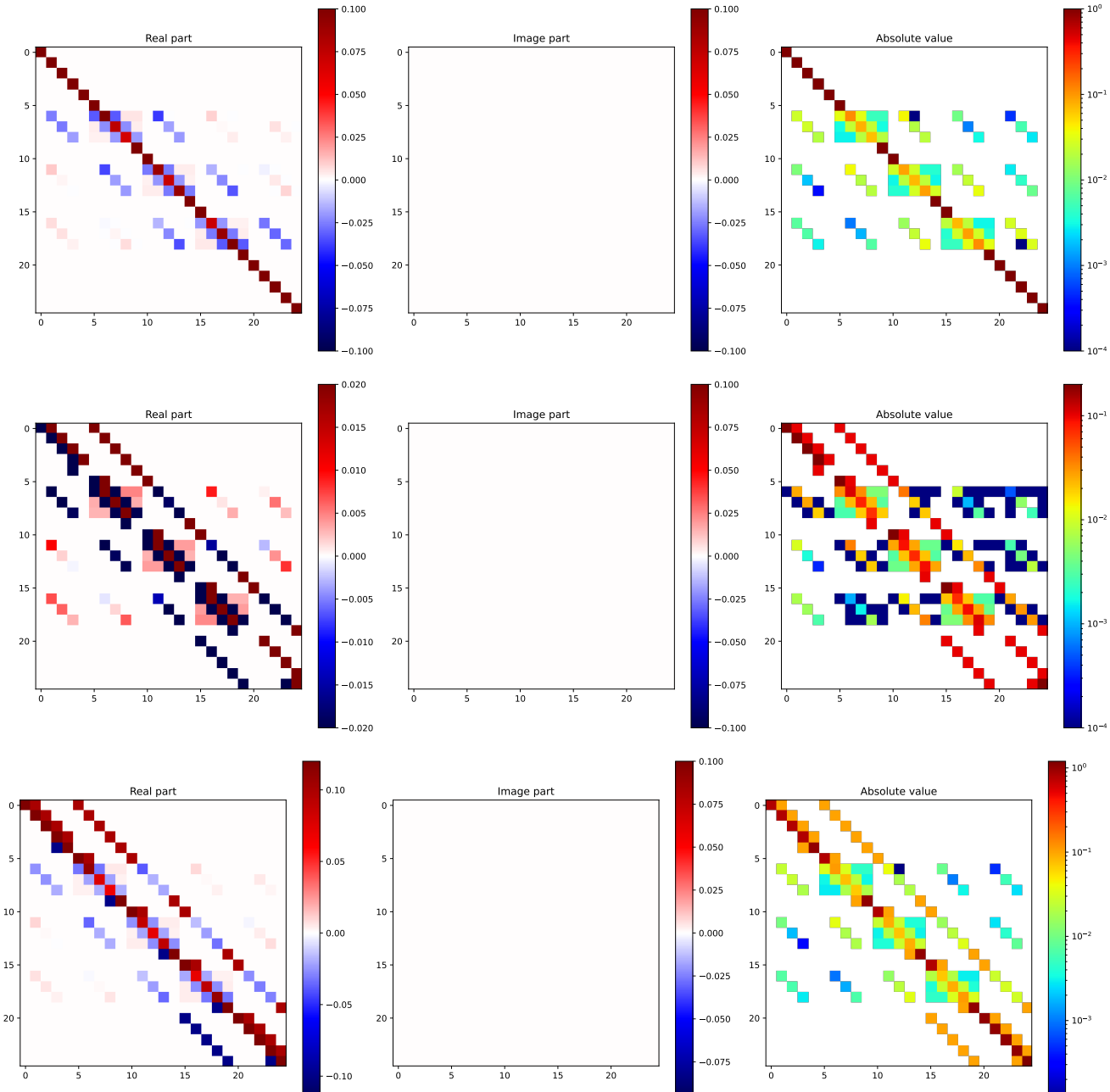
Visualization of three 25×25 FDM coefficient matrices



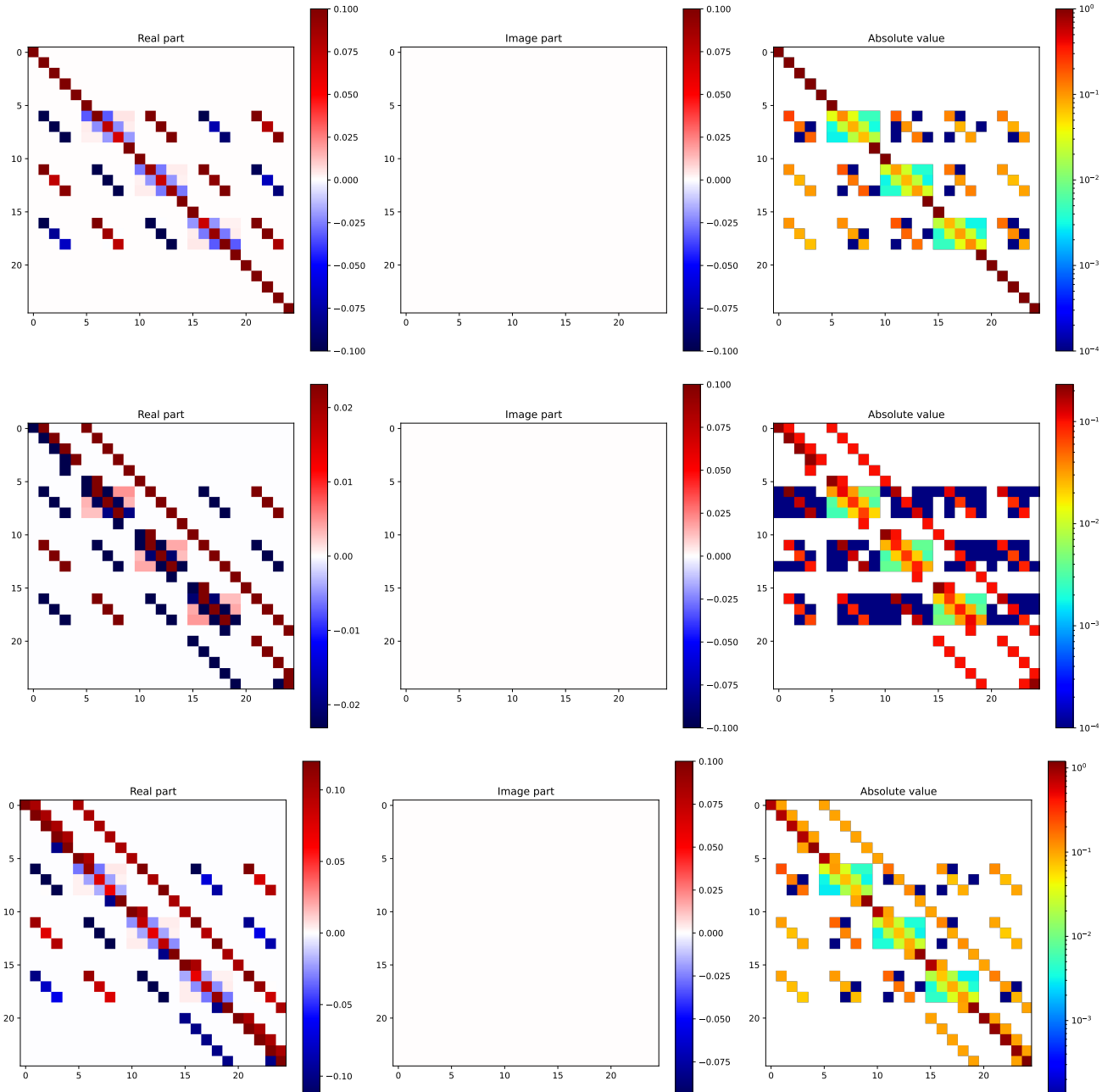
Visualization of three 25×25 FFT coefficient matrices



Visualization of three 25×25 FFT coefficient matrices



Visualization of three 25×25 FFT coefficient matrices



Bibliography I

0 [OL Survey] Nicolas Boullé, Alex Townsend. A Mathematical Guide to Operator Learning. *arXiv*, 45 pages, 2023, https://doi.org/10.48550/arXiv.2312.14688

1 [FGMRES] Y. Saad. A Flexible Inner-Outer Preconditioned GMRES Algorithm. *SIAM Journal on Scientific Computing*, 14(2), 1993, DOI: 10.1137/0914028

2 [Fluid Simulation] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating Eulerian Fluid Simulation with Convolutional Networks. *In Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3424–3433. PMLR, 2017, https://proceedings.mlr.press/v70/tompson17a.html

3 [PINNs] L. Lu, R. Pestourie, W.-J. Yao, Z.-C. Wang, F. Verdugo, and S. G. Johnson. Physics- Informed Neural Networks with Hard Constraints for Inverse Design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021, DOI: 10.1137/21M1397908

4 [DeepXDE] L. Lu, X.-H. Meng, Z.-P. Mao, and G. E. Karniadakis. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review*, 63(1):208–228, 2021, DOI: 10.1137/19M1274067

5 [Theory in DNN] B. Adcock and N. Dexter. The Gap between Theory and Practice in Function Approximation with Deep Neural Networks. *SIAM Journal on Mathematics of Data Science*, 3(2):624-655, 2021, DOI: 10.1137/20M131309X

6 [OL with DeepONet] A. Kopaničáková and G. Em Karniadakis. DeepOnet Based Preconditioning Strategies For Solving Parametric Linear Systems of Equations. *arXiv*, 2024, DOI: 10.48550/arXiv.2401.02016

7 [U-Net] O. Ronneberger, F. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *In Lecture Notes in Computer Science*, Springer International Publishing, 2015, DOI: 10.1007/978–3–319–24574–4_28