Digital Twins in Engineering & Artificial Intelligence and Computational Methods in Applied Science,
DTE - AICOMAS 2025
February 17-21, 2025, Paris, France

CONVOLUTION NEURAL OPERATOR PRECONDITIONING FOR THE SOLUTION OF SOME HETEROGENEOUS PDES

Yanfei Xiang* and Luc Giraud†

* Concace, Inria project with Airbus CR & T and Cerfacs, France
CERFACS, 42 Av. Gaspard Coriolis, 31100, Toulouse, France
yfxiang0amber@gmail.com and https://www.linkedin.com/in/yanfei-xiang-b2a461209/?originalSubdomain=fr

† Concace, Inria project with Airbus CR & T and Cerfacs, France
CERFACS, 42 Av. Gaspard Coriolis, 31100, Toulouse, France
luc.giraud@inria.fr and https://concace.gitlabpages.inria.fr/members/luc-giraud/

Key Words: *Heterogeneous equations, Poisson equations, Darcy flow, Diffusion-Advection equations, Scientific machine learning, Convolution neural networks, Unsupervised operator learning, Subspace iterative methods on CPUs and/or on GPUs.*

ABSTRACT

This work exclusively focuses on using convolution neural operator learning for accelerating the solution of some heterogenous PDEs (including Poisson equations, Darcy flow, Diffusion-Advection equations) using the flexible GMRES [1] method. We use operator learning with U-Net [2] neural network architecture. For the sake of learning general information, the neural operator is trained with randomly generated datasets using an unsupervised approach. The trained neural operator exhibits significant generalization features with respect to different aspects. That includes the ability to address varying source terms, diffusivity terms, velocity field for advection, and varying boundary conditions for these heterogeneous equations. Furthermore, it also shows promising results for addressing wider range of the advection dominant situations, which is the challenging case for the advection-diffusion equations. Overall, this work demonstrates the efficiency of applying the neural operator learning as felxible preconditioner for subspace iterative linear solvers.

REFERENCES

[1] Y. Saad, A Flexible Inner-Outer Preconditioned GMRES Algorithm. SIAM Journal on Scientific Computing. 1993

[2] 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.

Unsupervised convolution neural operator preconditioning for the solution of some heterogeneous PDEs

MS044B - DTE & AICOMAS 2025, Paris, France

February 21, 2025

Yanfei Xiang (✉ yfxiang0amber@gmail.com)

Concace team, Centre Inria de l'Université de Bordeaux

Some heterogeneous fluid equations

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:

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

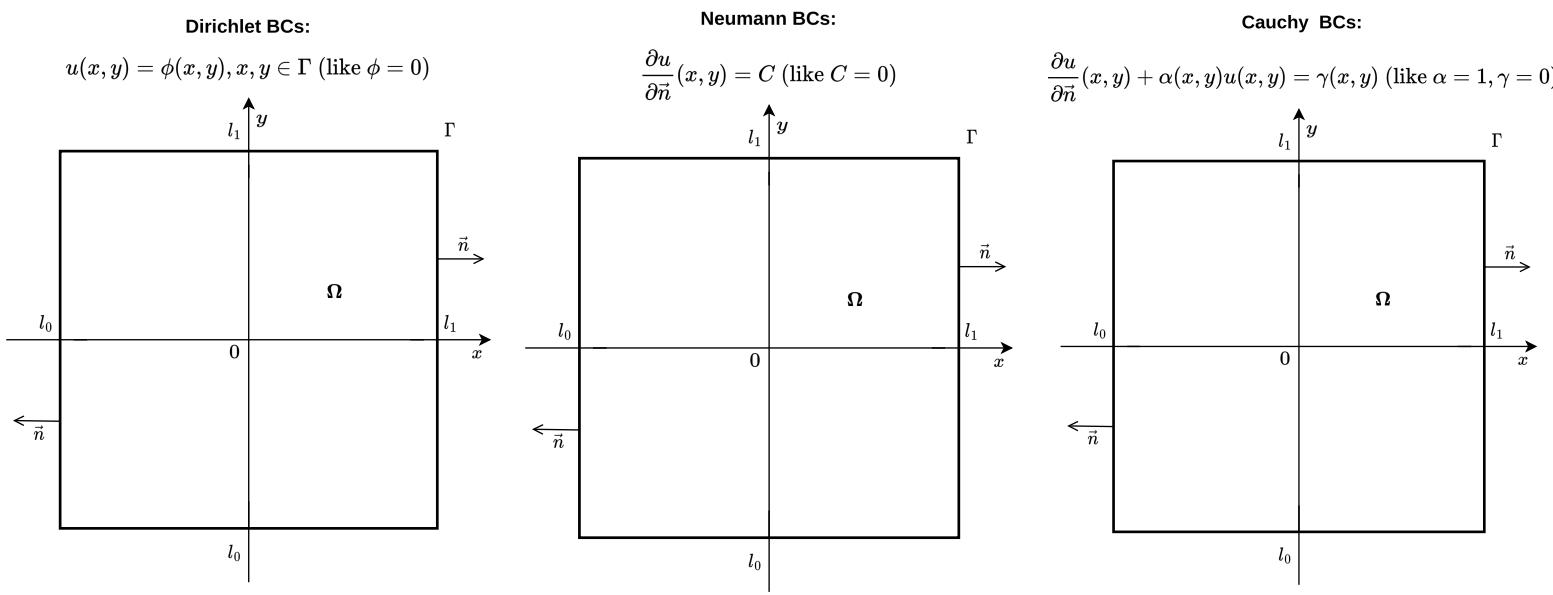
- let $x \in \mathbb{R}^d$ be the grid points in the d-dimensional Ω ($d = 1, \mathbf{2}, 3$);
- $u(x) : \mathbb{R}^d \to \mathbb{R}$ is the solution to be computed;
- $\rho(x) : \mathbb{R}^d \to \mathbb{R}$ is a parametric source/forcing term;
- $p(x) : \mathbb{R}^d \to \mathbb{R}_+$ represents the heterogeneous permeability of the porous medium (non-uniform $p(x)$ varies from point to point);
- $d(x) : \mathbb{R}^d \to \mathbb{R}_+$ denotes the heterogeneous diffusivity or viscosity field associated with diffusion;
- $c(x) : \mathbb{R}^d \to \mathbb{R}_+$ represents the heterogeneous velocity or force 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:



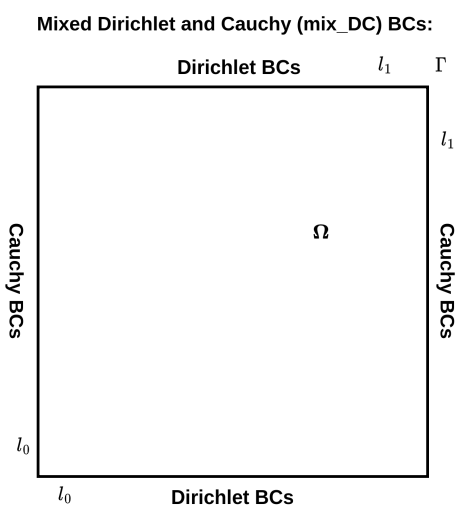
(a). Dirichlet BCs (b). Neumann BCs (c). Cauchy BCs

Varying Boundary Conditions (BCs)

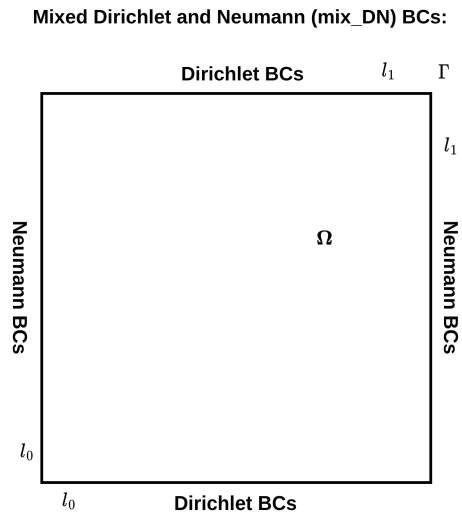
For the heterogeneous fluid PDEs described in Equation (1)-(3), we consider three common classical BCs, Dirichlet, Neumann, and Cauchy:

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

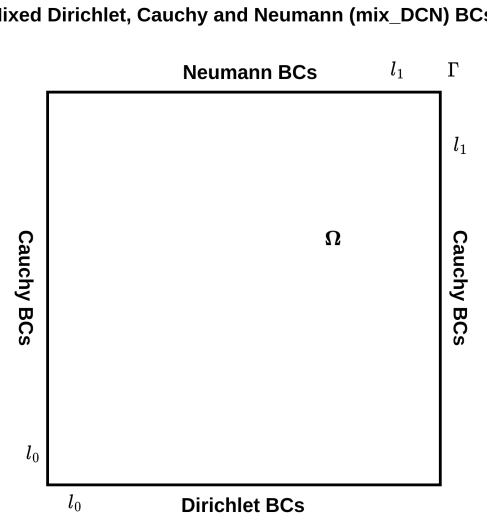
Visualize their mixtures, i.e., the combinations across different boundary:



(*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:

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

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:

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

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

- are usually costly in training[b], and may fail to solve 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:

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

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

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

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**.

* $\mathcal{F}_\theta$: CNNs (Convolution NNs) with U-Net architecture[a] with 4 depth
* **Loss-function:**

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

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

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 FGMRES case, we have

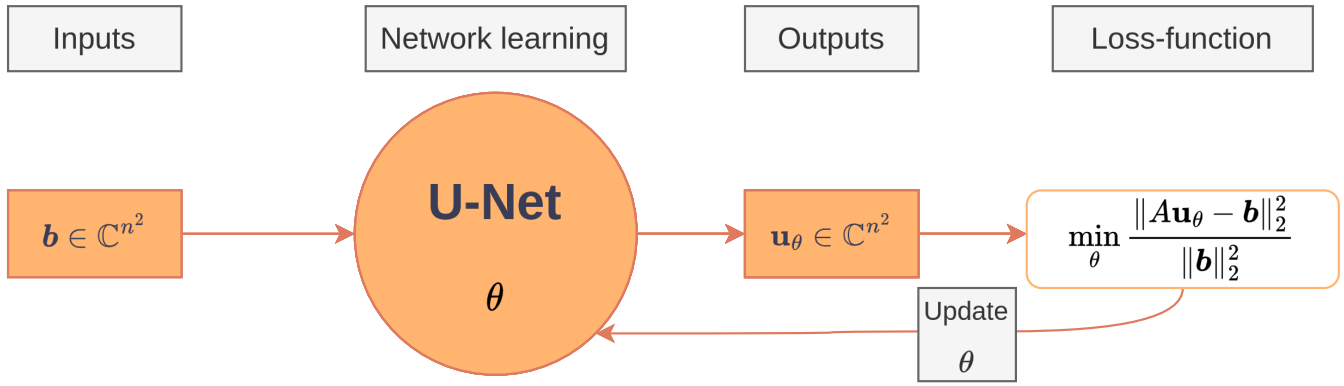
$$z_j = \mathcal{F}_\theta([v_j, *, *]),\ j = 1, \ldots m, \tag{8}$$

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

Neural operator preconditioning for Poisson eqs.



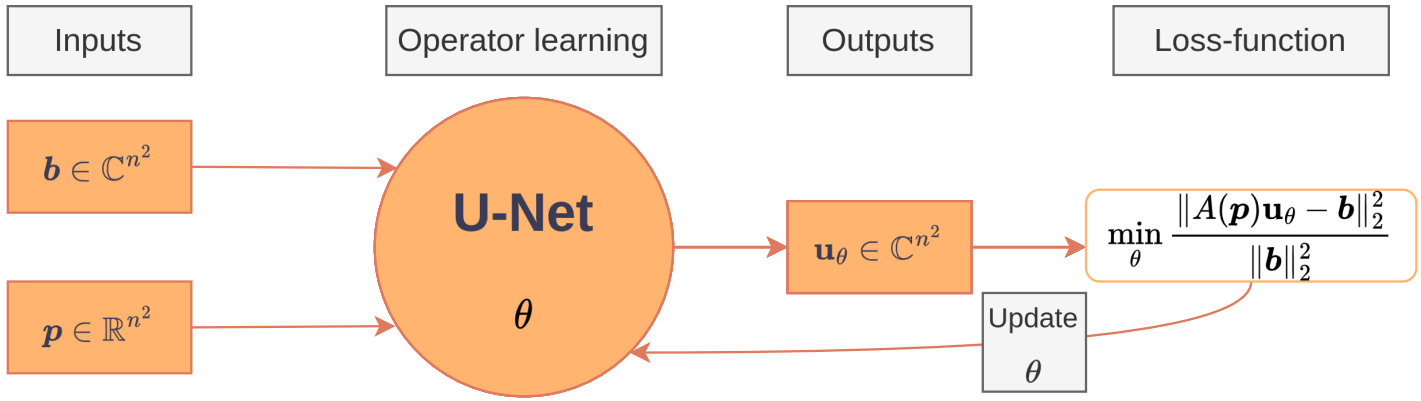
Training (on 2 V100 GPUs, depth $= 4$, train. time: 1.28 hour):

- 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
- * Training with zero-Dirichlet BCs is suffice to obtain effective preconditioning operator
- * Discrete with FFT instead of FDM: Preconditioning operator leverages the statistical properties of fluid data and captures the local sparsity structure of the discrete equations
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 831 K

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



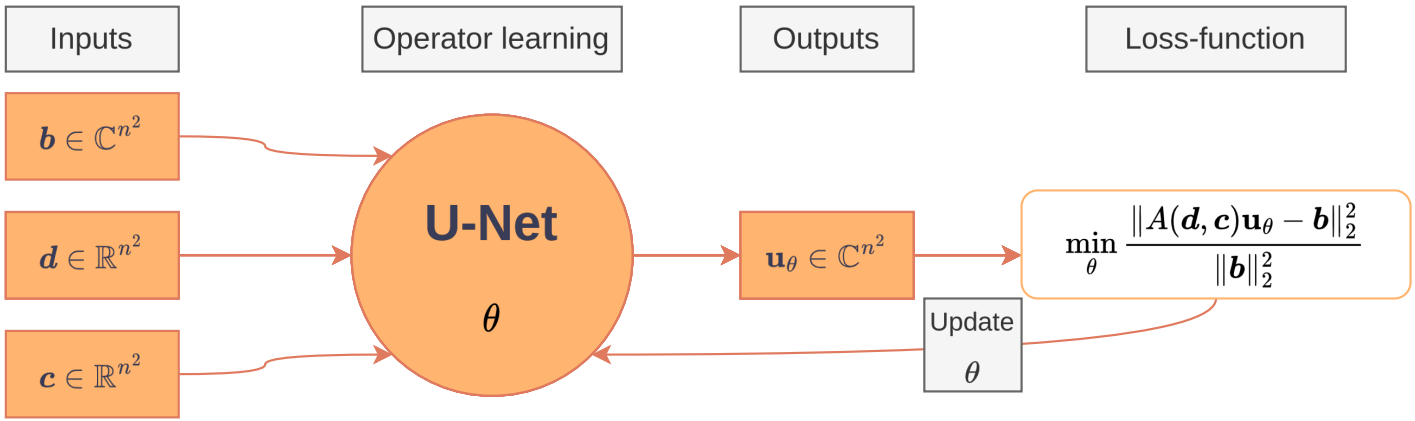
Training (on 2 V100 GPUs, depth $= 4$, train. time: 1.33 hour):

- 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 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 2 V100 GPUs, depth $= 4$, train. time: 1.44 hour):

- 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 $\in \mathbb{R}^{n^2+} \sim \mathcal{U}(1, 2)$
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 831 K

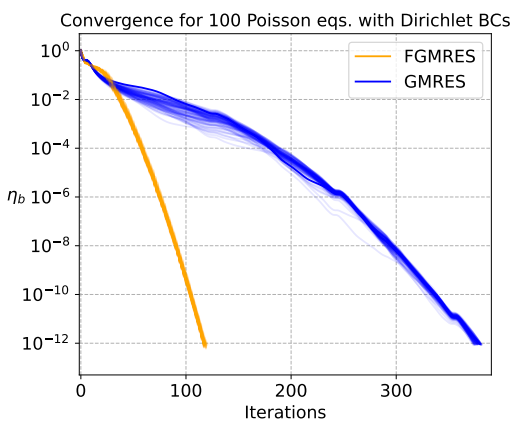
Testing:

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

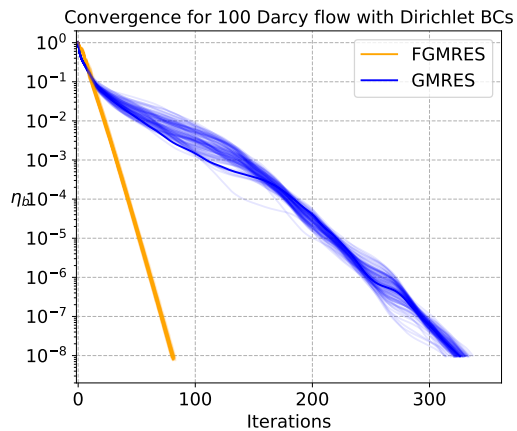
General testing results

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:

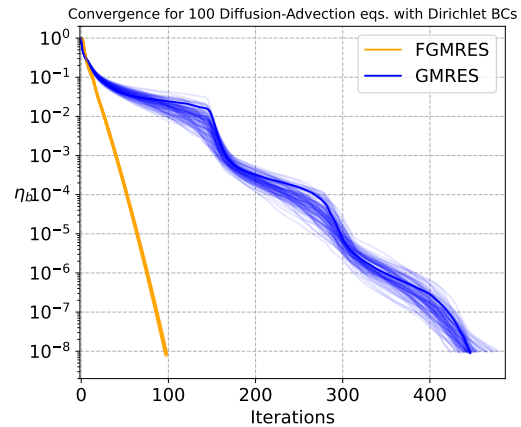
- Stopping criterion: $\eta_b = \frac{\|A\mathbf{u} - \mathbf{b}\|}{\|\mathbf{b}\|} \leq \varepsilon$ with
  $\varepsilon = 10^{-12}$ for Poisson eqs. (1);
  $\varepsilon = 10^{-8}$ for Darcy flows (2) and Convection-Diffusion eqs. (3);
- Maximum dimension of the Krylov search space is set to be 512; number of restarts is set to be 10. Thus the whole maximum iteration is 5120;
- 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 $= 467$,
val_loss $= 0.05100$



Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$,
with epoch $= 401$,
val_loss $= 0.15009$

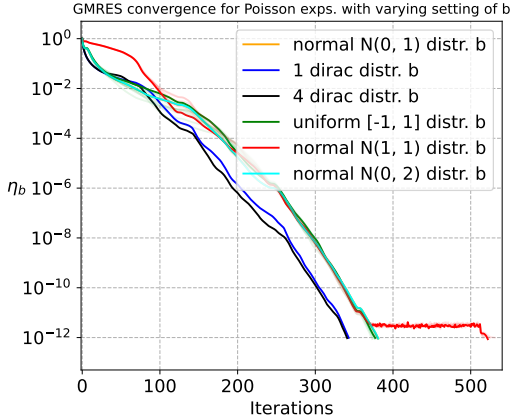


Convection-Diffusion eqs.:
$A([\mathrm{d}, \mathrm{c}])\mathrm{u} = \mathrm{b}$, with
epoch $= 483$,
val_loss $= 0.09784$

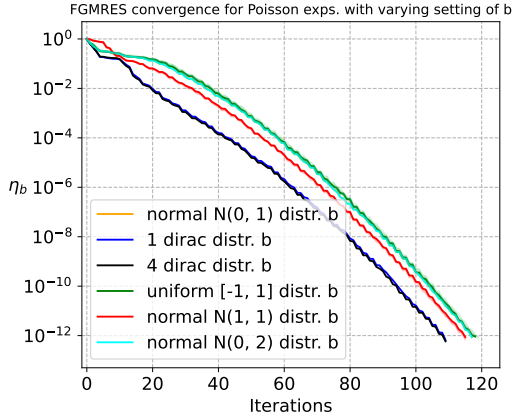
1. Varying the source term b

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

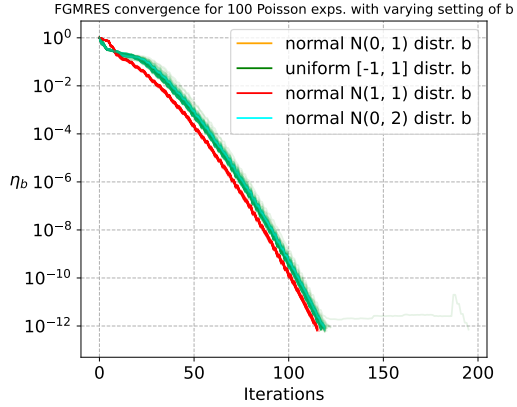
Results of parametric Poisson eqs.: $A\text{u} = \text{b}$



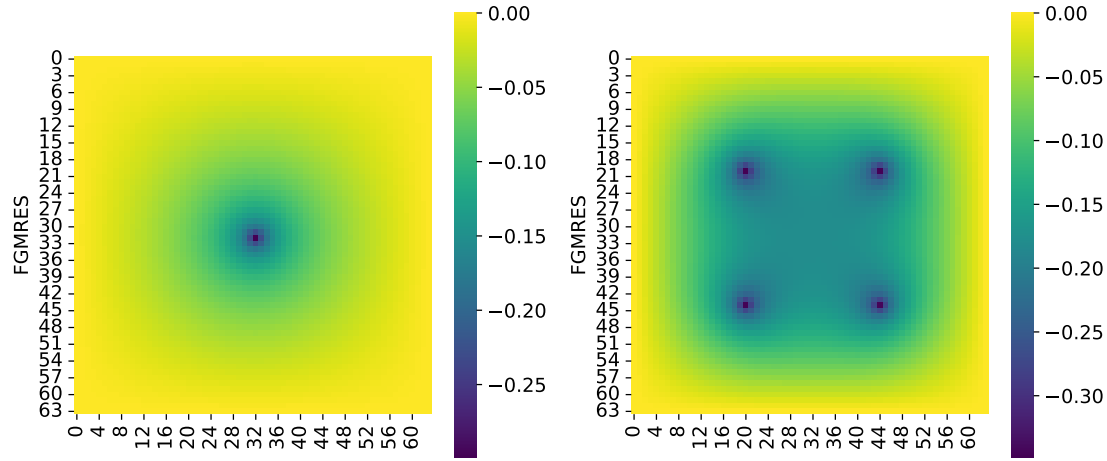
(*a*). 5 GMRES results



(*b*). 5 FGMRES results



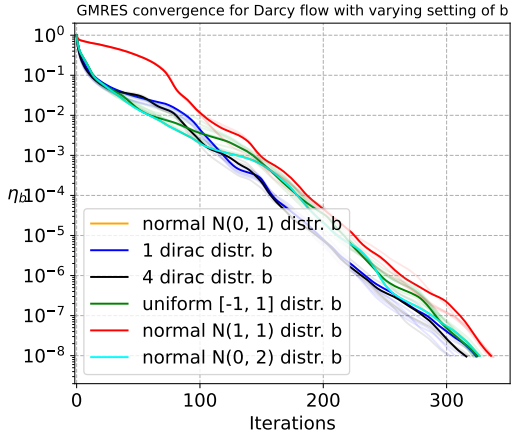
(*c*). 100 FGMRES results



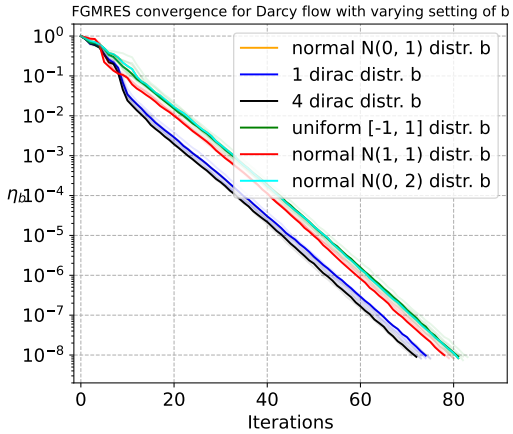
1. Varying the source term b

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:

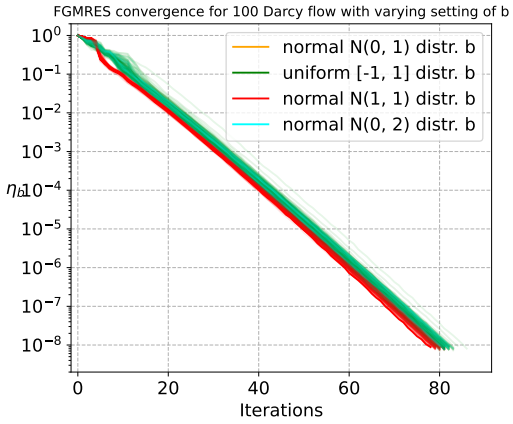
Results of heterogeneous Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$



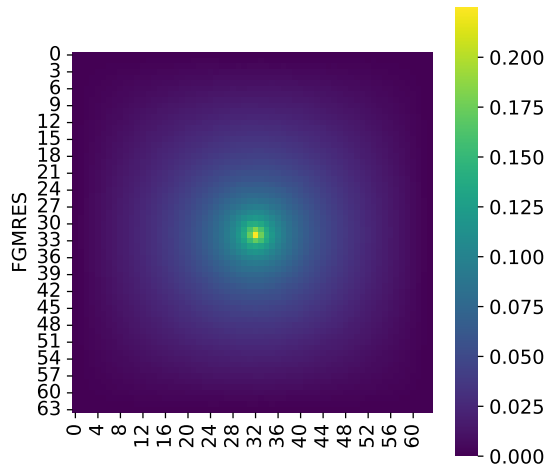
(*d*). 5 GMRES results

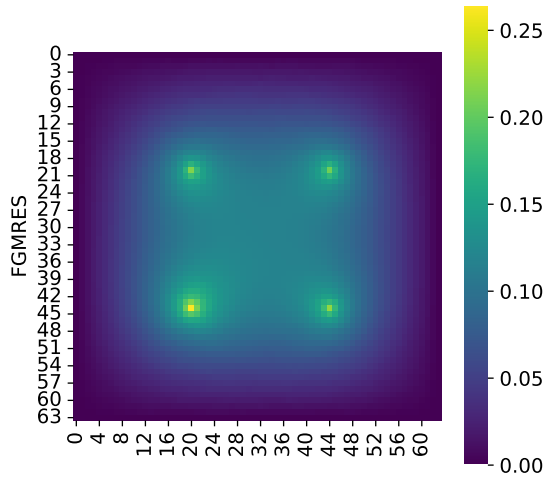


(*e*). 5 FGMRES results



(*f*). 100 FGMRES results

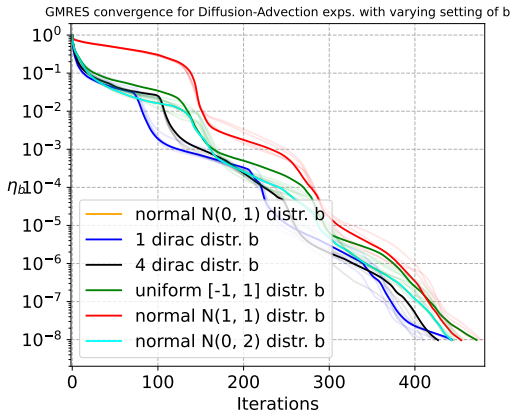




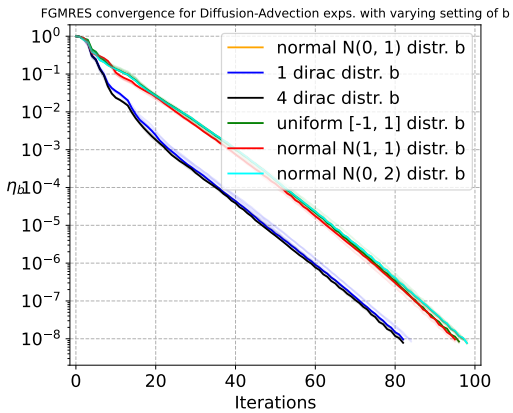
1. Varying the source term b

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:

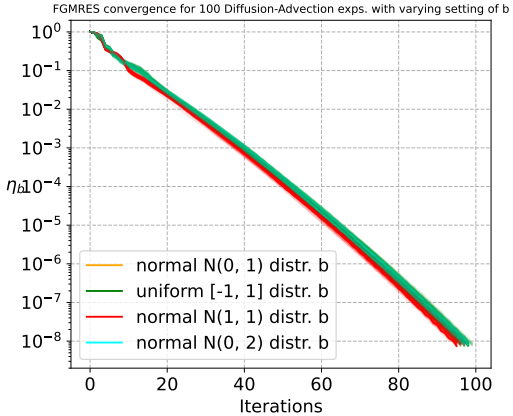
Results of heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}])\mathrm{u} = \mathrm{b}$



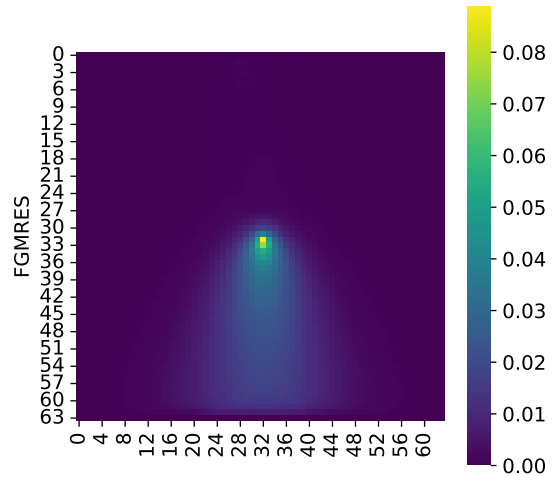
(g). 5 GMRES results

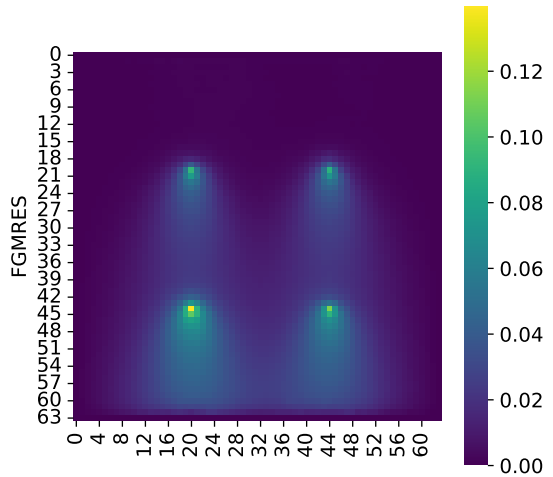


(h). 5 FGMRES results



(i). 100 FGMRES results

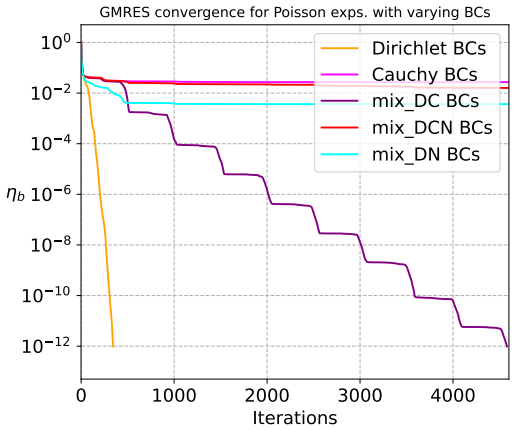


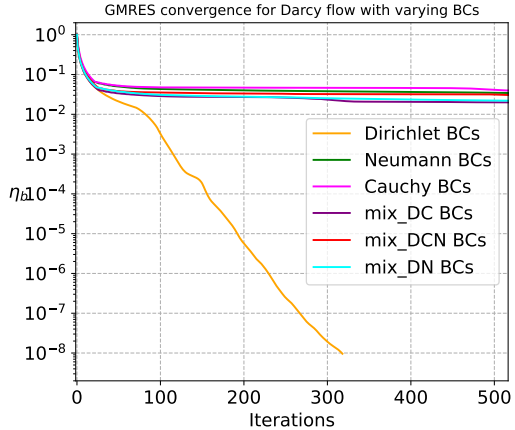


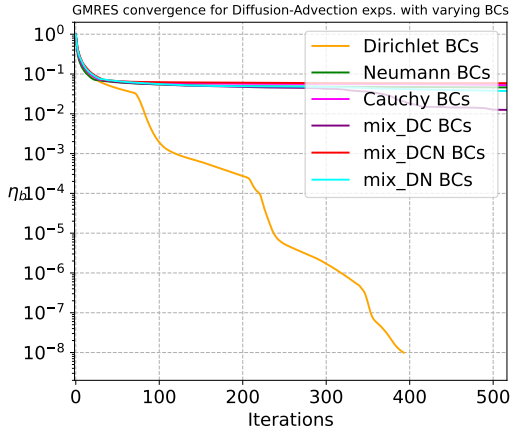
2. Varying the boundary conditions (BCs)

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 1 `dirac distr.` b:

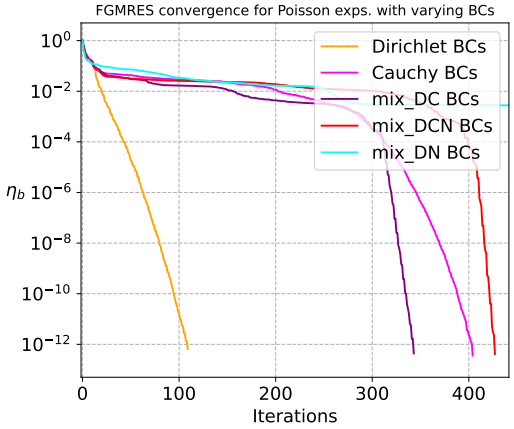
(a). GMRES results

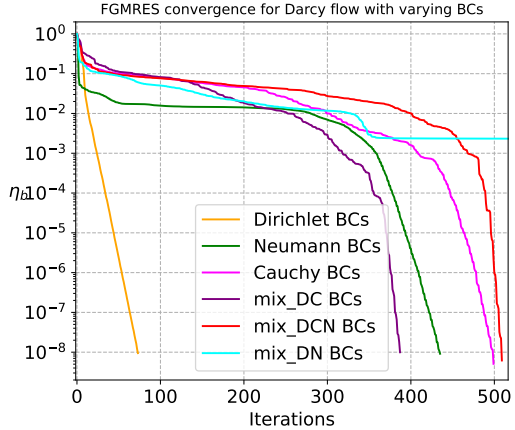


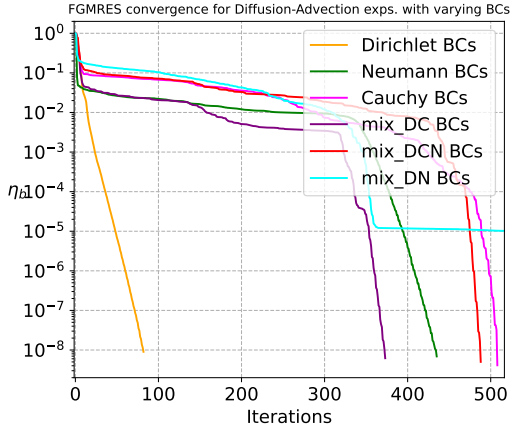




(b). FGMRES results





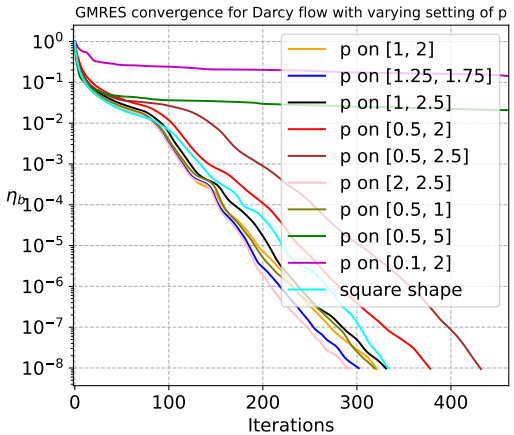


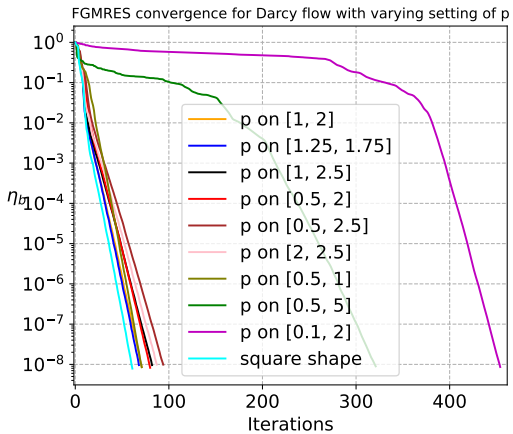
Poisson eqs. | Darcy flows | Convection-Diffusion eqs.

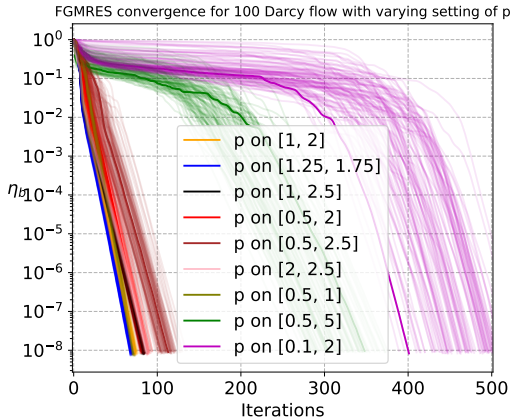
3. Varying the permeability or diffusion term

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 1 `dirac distr.` b:

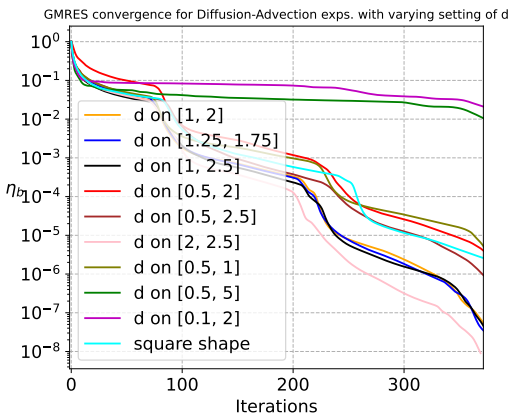
Heterogeneous Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$

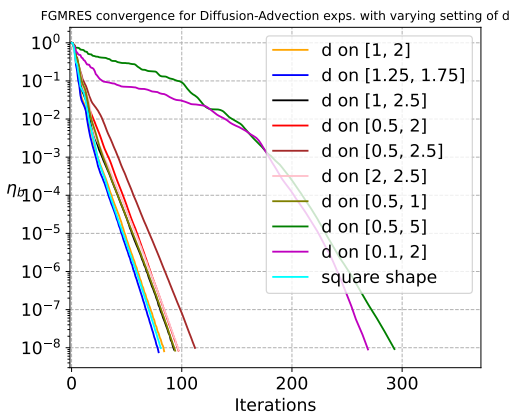


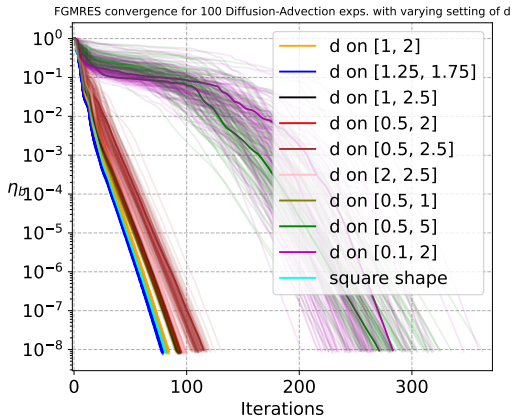




Heterogeneous Convection-Diffusion eqs.: $A([\mathrm{d}, \mathrm{c}])\mathrm{u} = \mathrm{b}$

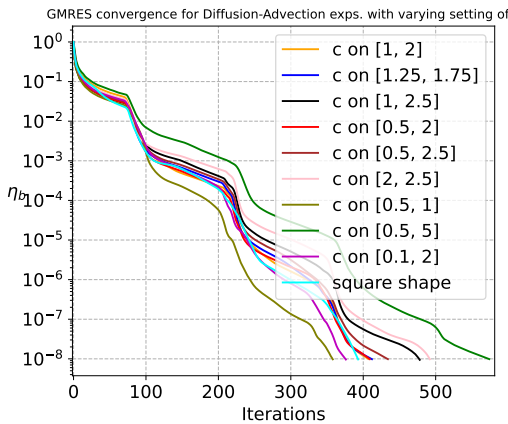


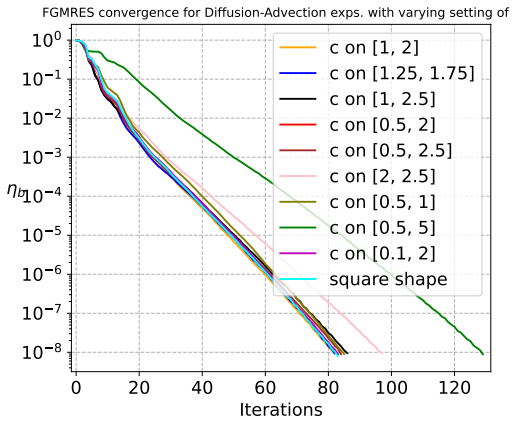


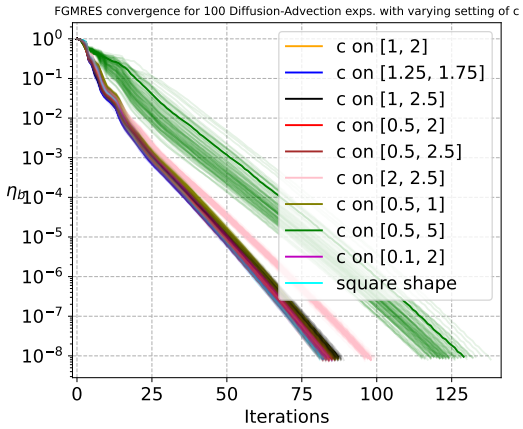


4. Varying the convection term

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 1 `dirac distr.` 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 1 `dirac distr.` b:

# $\mathbf{d}, \mathbf{c} \in \mathbb{R}^{64^2}$ on	Method	L	its	ET
Balanced case with $\frac{\mathbf{d}}{\mathbf{c}} \approx 1$:				
$\mathbf{d}, \mathbf{c} \in [1, 2]$	GMRES	1	408	35.5709s
	FGMRES	1	85	3.3399s
$\mathbf{d}, \mathbf{c} \in [25, 50]$	GMRES	1	420	50.2798s
	FGMRES	1	233	17.5740s
$\mathbf{d}, \mathbf{c} \in [50, 100]$	GMRES	1	460	54.7812s
	FGMRES	1	250	20.2132s
$\mathbf{d}, \mathbf{c} \in [100, 200]$	GMRES	1	468	57.3766s
	FGMRES	1	283	24.2253s
Diffusion-dominant case with $\frac{\mathbf{d}}{\mathbf{c}} \gg 1$:				
$\mathbf{d} \in [25, 50], \mathbf{c} \in [1, 2]$	GMRES	1	325	21.8579s
	FGMRES	1	276	16.8212s
$\mathbf{d} \in [50, 100], \mathbf{c} \in [1, 2]$	GMRES	1	325	30.7067s
	FGMRES	1	302	30.4414s
Convection-dominant case with $\frac{\mathbf{d}}{\mathbf{c}} \ll 1$:				
$\mathbf{d} \in [1, 2], \mathbf{c} \in [25, 50]$	GMRES	1	2854	283.7008s
	FGMRES	1	1021	107.3359s
$\mathbf{d} \in [1, 2], \mathbf{c} \in [50, 100]$	GMRES	1	4436	597.5046s
	FGMRES	1	3937	374.7240s

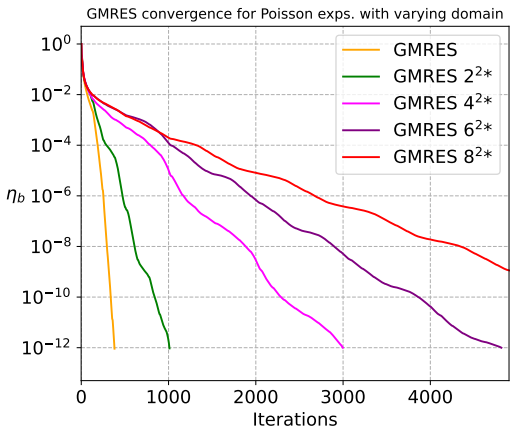
Three cases for the Convection-Diffusion eqs.:

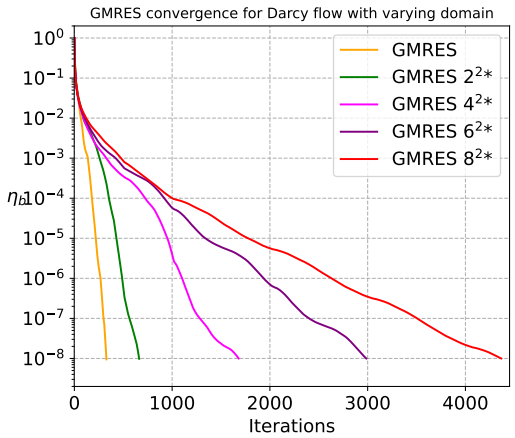
- **Balanced case** with $\frac{\mathbf{d}}{\mathbf{c}} \approx 1$: convection and diffusion effects are balanced
- **Diffusion-dominant case** with $\frac{\mathbf{d}}{\mathbf{c}} \gg 1$: diffusion significantly outweighs convection
- **Convection-dominant case** with $\frac{\mathbf{d}}{\mathbf{c}} \ll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

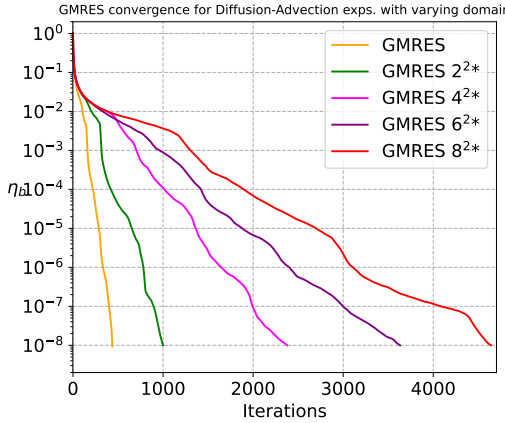
5. Varying the domain size

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*):

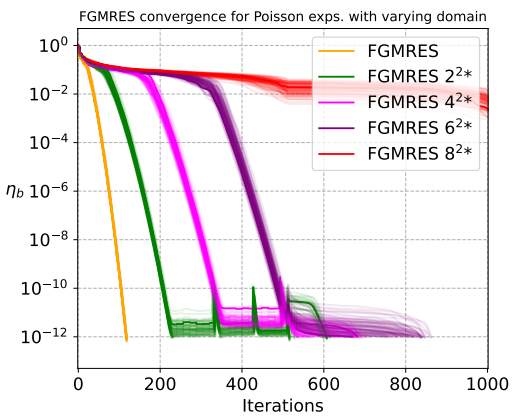
(a). GMRES results

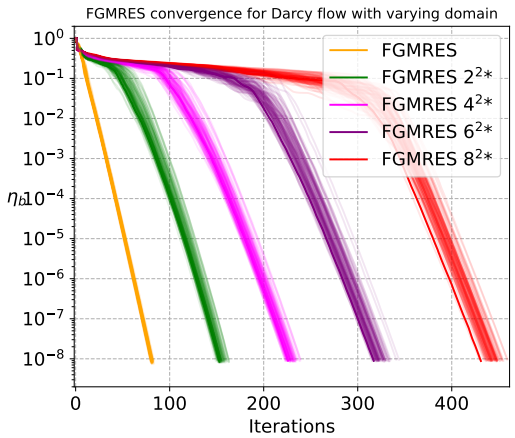


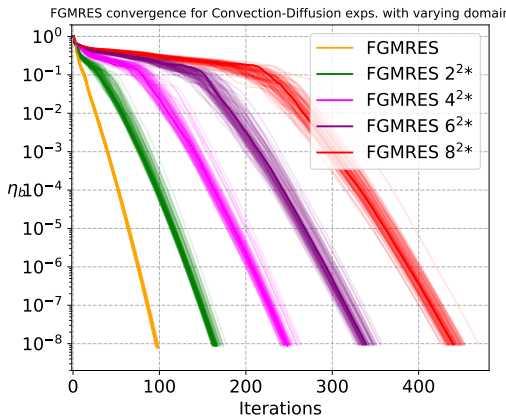




(b). 100 FGMRES results







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 heterogeneous fluid equations.

Machine learning + iterative methods 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:

- heterogeneous fluid equations with varying parameters; with varying boundary conditions;
  - Convection-Diffusion equations with even larger range (up to 100 times) of the convection and diffusion terms; and apply to the challenging Convection-dominant cases.

Research report version of this work is accessible online at Inria-HAL (https://inria.hal.science/hal-04886933v2).

$\longrightarrow$ **Yanfei Xiang (yfxiang0amber@gmail.com)**

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.*** Fahy et al., 2024, Li et al., 2024, Azulay & Treister, 2022, Xiang 2022, Ackmann et al., 2021, Battaglia et al., 2018, Götz & Anzt, 2018, ...
- *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, ...

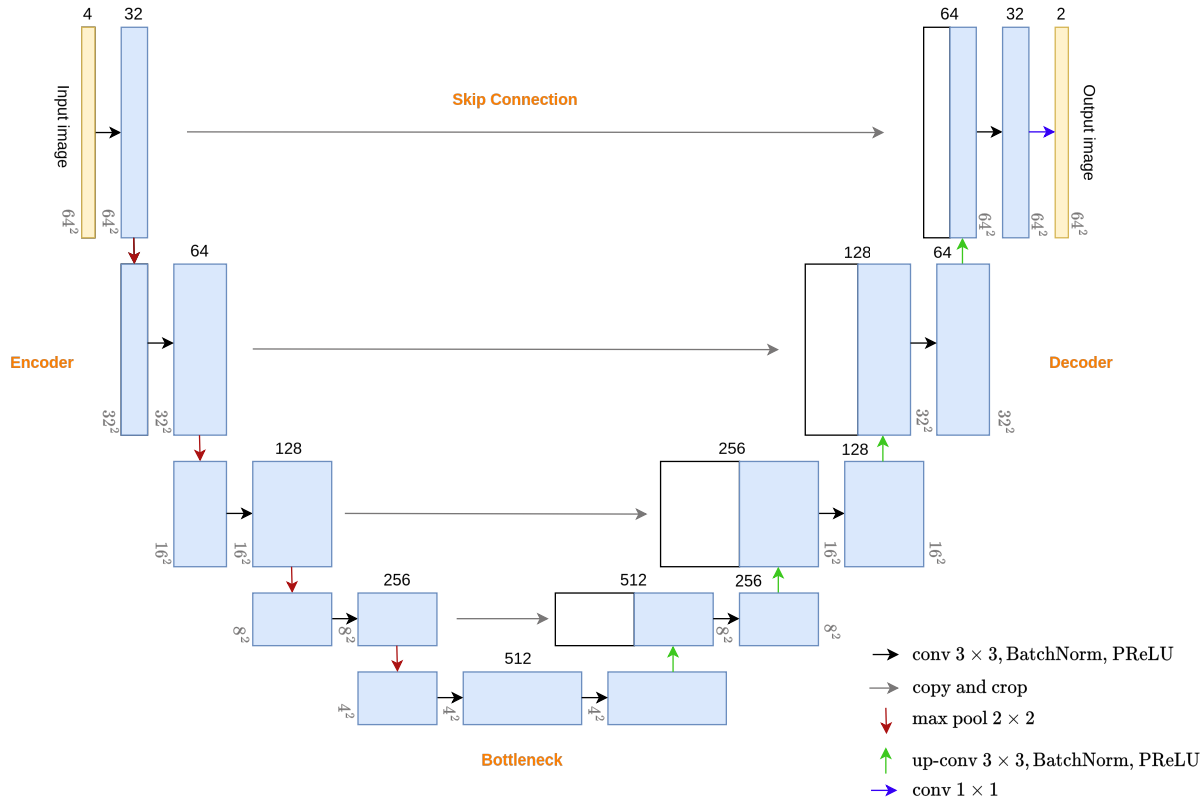
Registration and travel support for this presentation was provided by the Concace team, Centre Inria de l'Université de Bordeaux

Thank you for your attention!

Questions ?

U-Net architecture

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

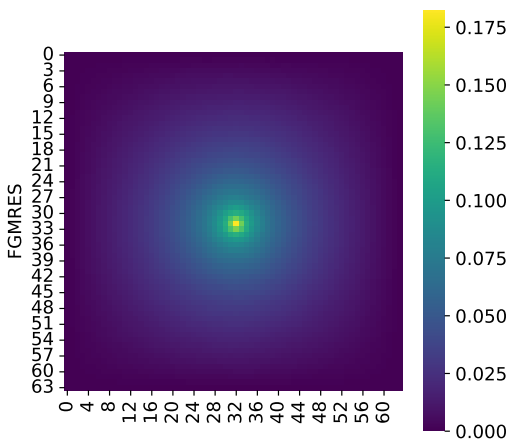


Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The *x*-*y*-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

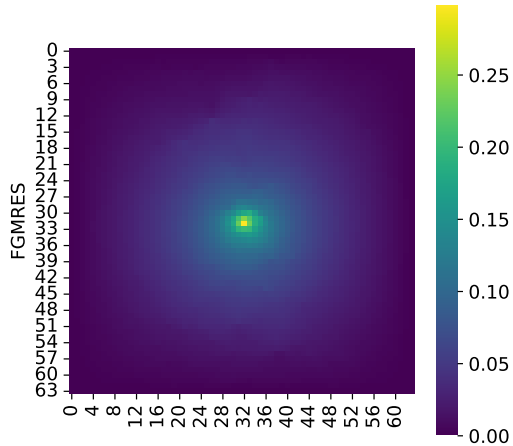
[a] Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015

Visualize Darcy flow with different permeability term p

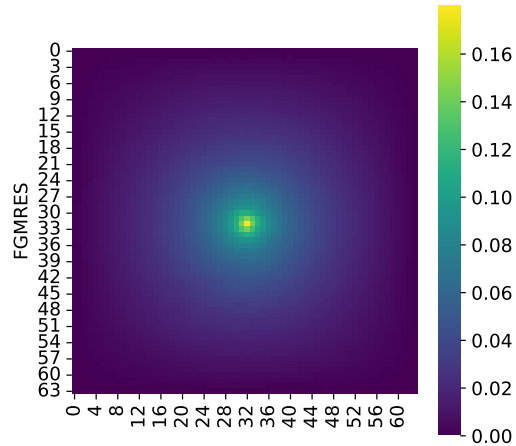
Heterogeneous Darcy flows: $A(\mathrm{p})\mathrm{u} = \mathrm{b}$



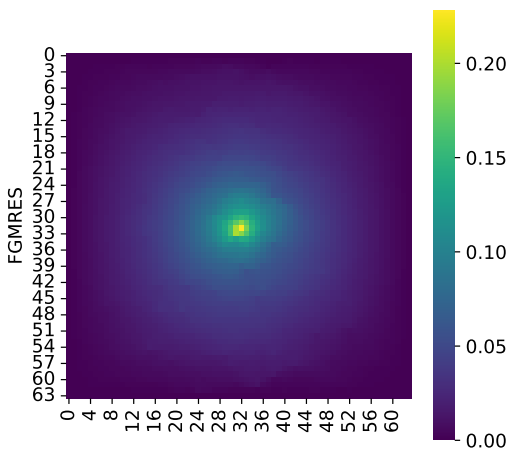
$\mathrm{p} \sim \mathcal{U}(1, 2)$



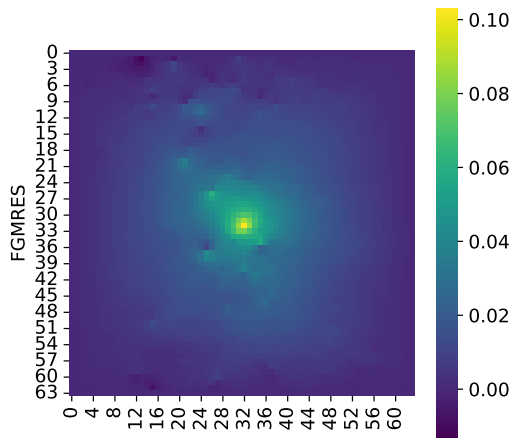
$\mathrm{p} \sim \mathcal{U}(0.5, 2)$



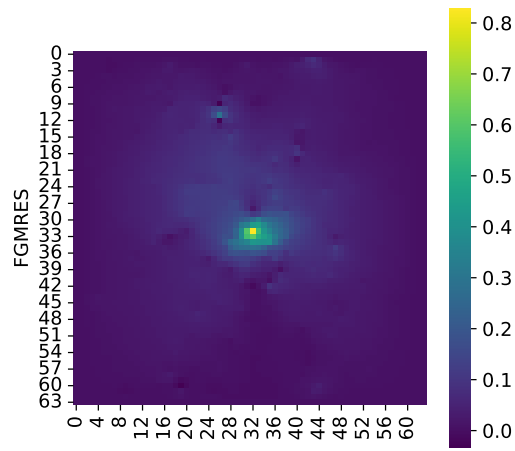
$\mathrm{p} \sim \mathcal{U}(1, 2.5)$



$\mathrm{p} \sim \mathcal{U}(0.5, 2.5)$

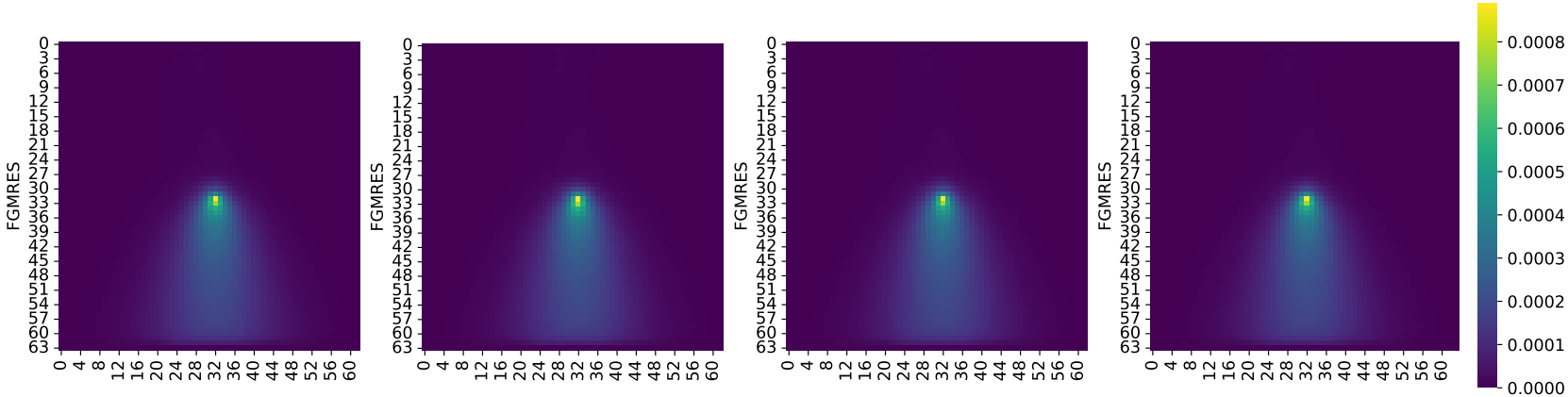


$\mathrm{p} \sim \mathcal{U}(0.5, 5)$

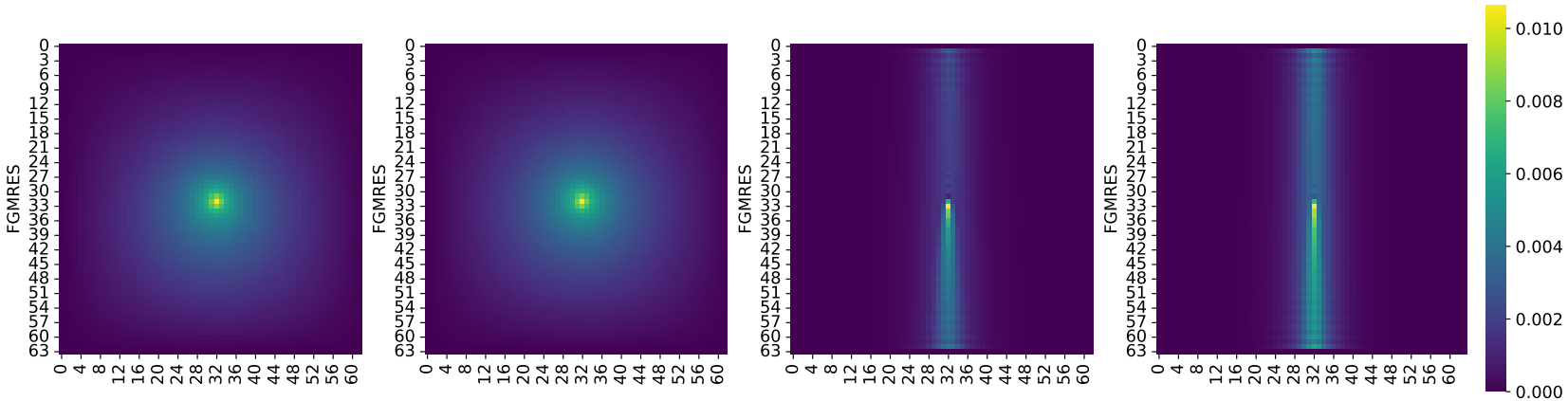


$\mathrm{p} \sim \mathcal{U}(0.1, 2)$

Visualize three cases of the Convection-Diffusion eqs.

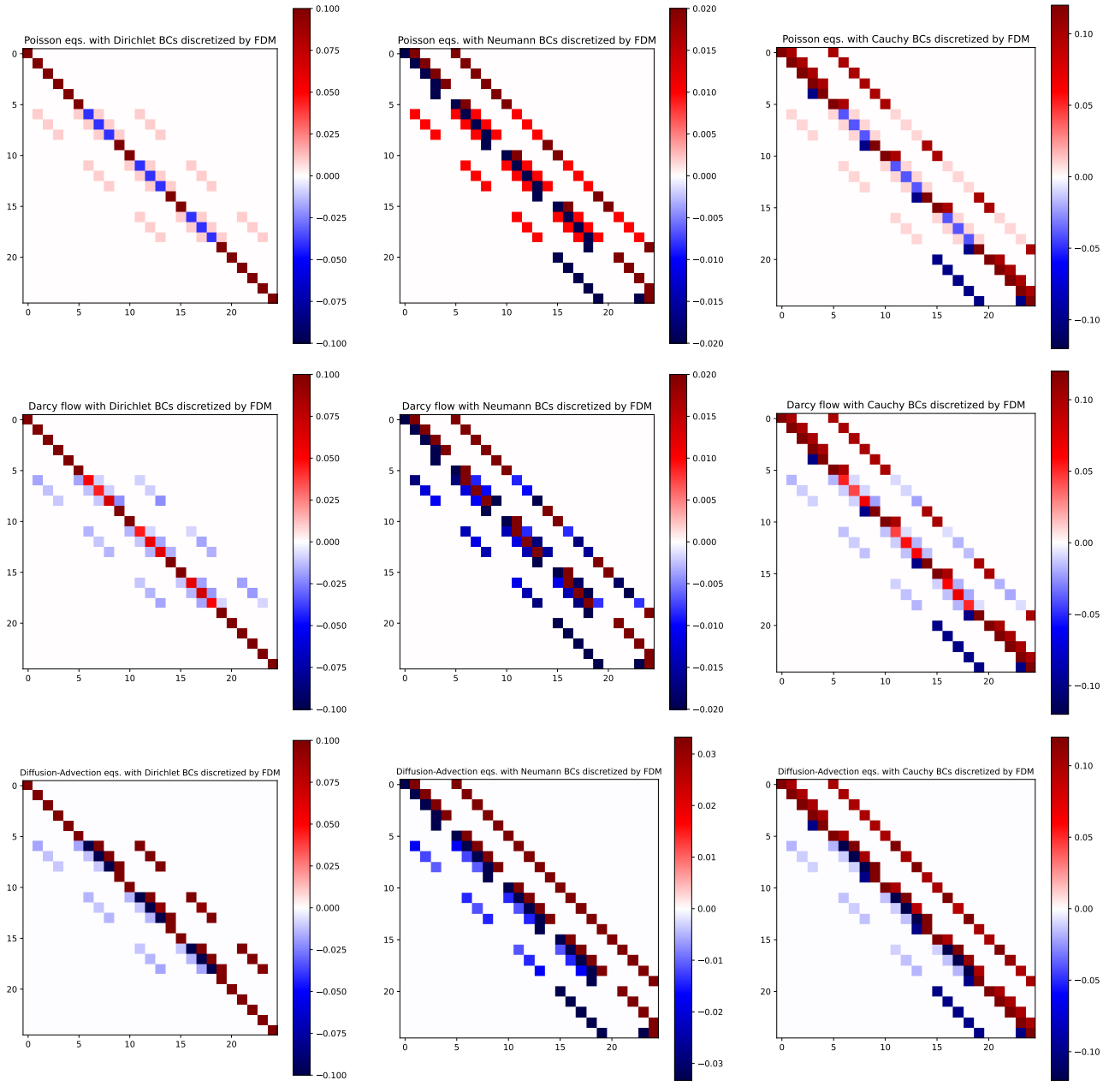


$\mathrm{d}, \mathrm{c} \in \mathcal{U}(1, 2)$ $\mathrm{d}, \mathrm{c} \in \mathcal{U}(25, 50)$ $\mathrm{d}, \mathrm{c} \in \mathcal{U}(50, 100)$ $\mathrm{d}, \mathrm{c} \in \mathcal{U}(100, 200)$

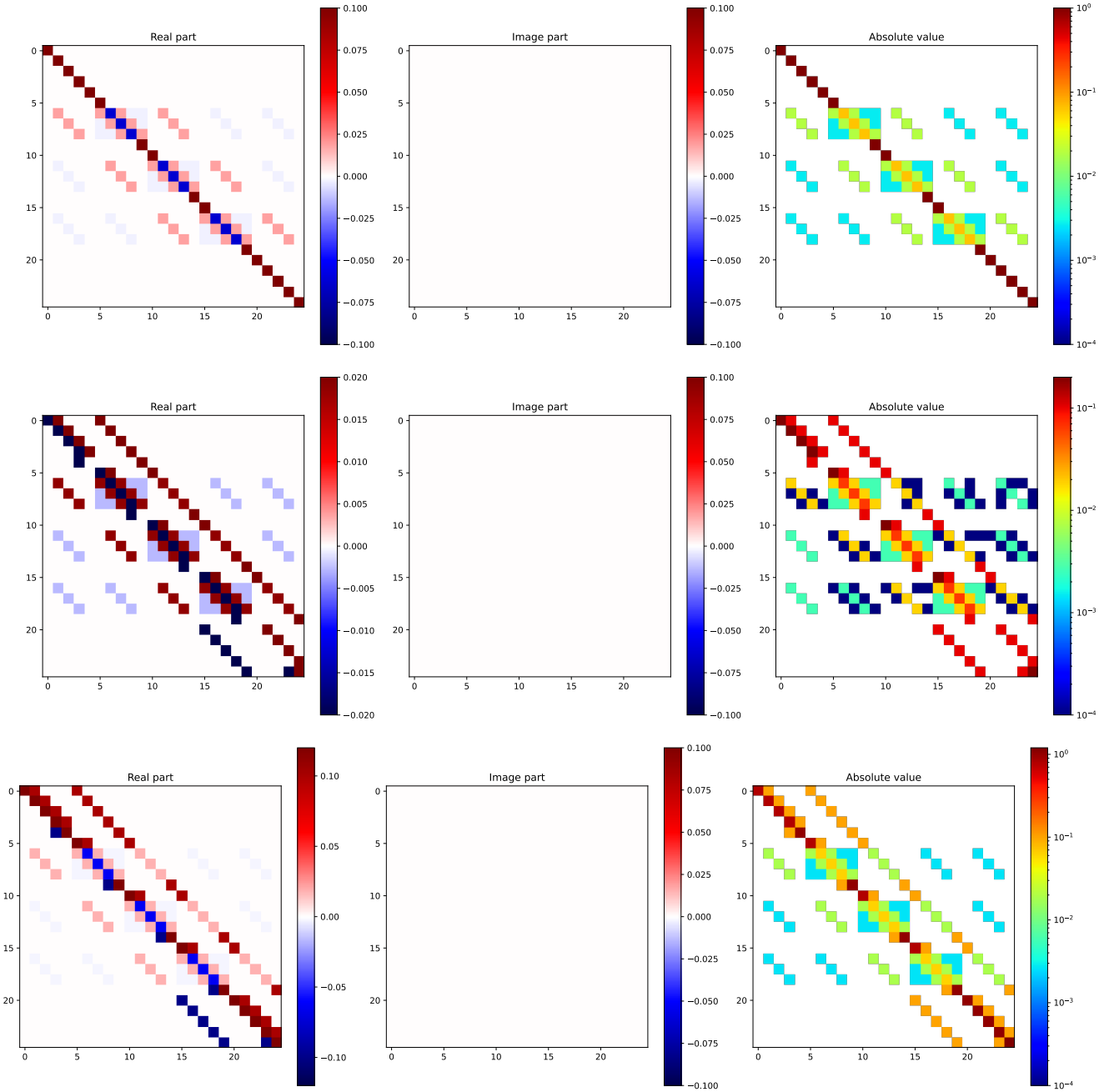


$\mathrm{d} \in \mathcal{U}(25, 50)$, $\mathrm{c} \in \mathcal{U}(1, 2)$ $\mathrm{d} \in \mathcal{U}(50, 100)$, $\mathrm{c} \in \mathcal{U}(1, 2)$ $\mathrm{d} \in \mathcal{U}(1, 2)$, $\mathrm{c} \in \mathcal{U}(25, 50)$ $\mathrm{d} \in \mathcal{U}(1, 2)$, $\mathrm{c} \in \mathcal{U}(50, 100)$

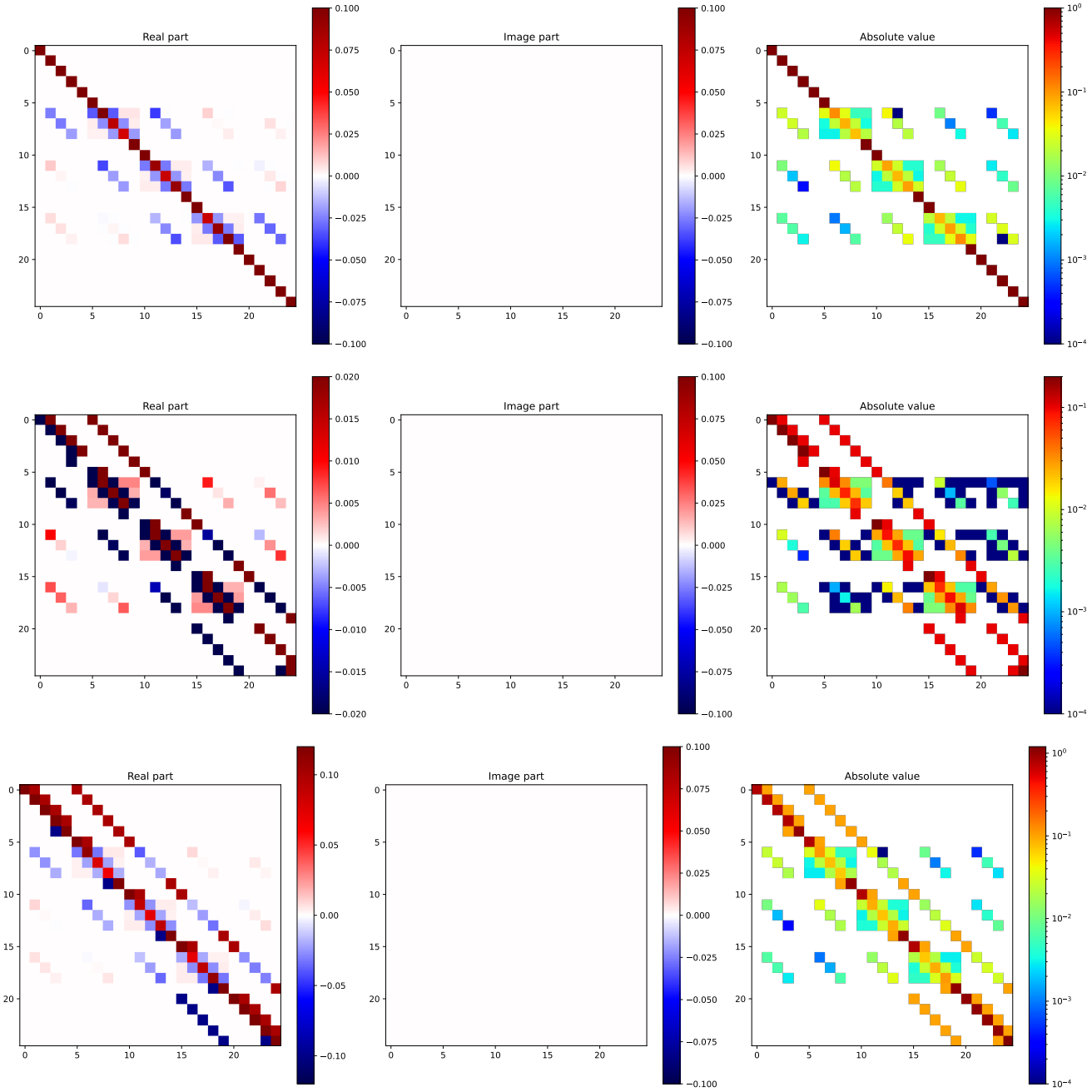
Visualization of three 25×25 FDM coefficient matrices



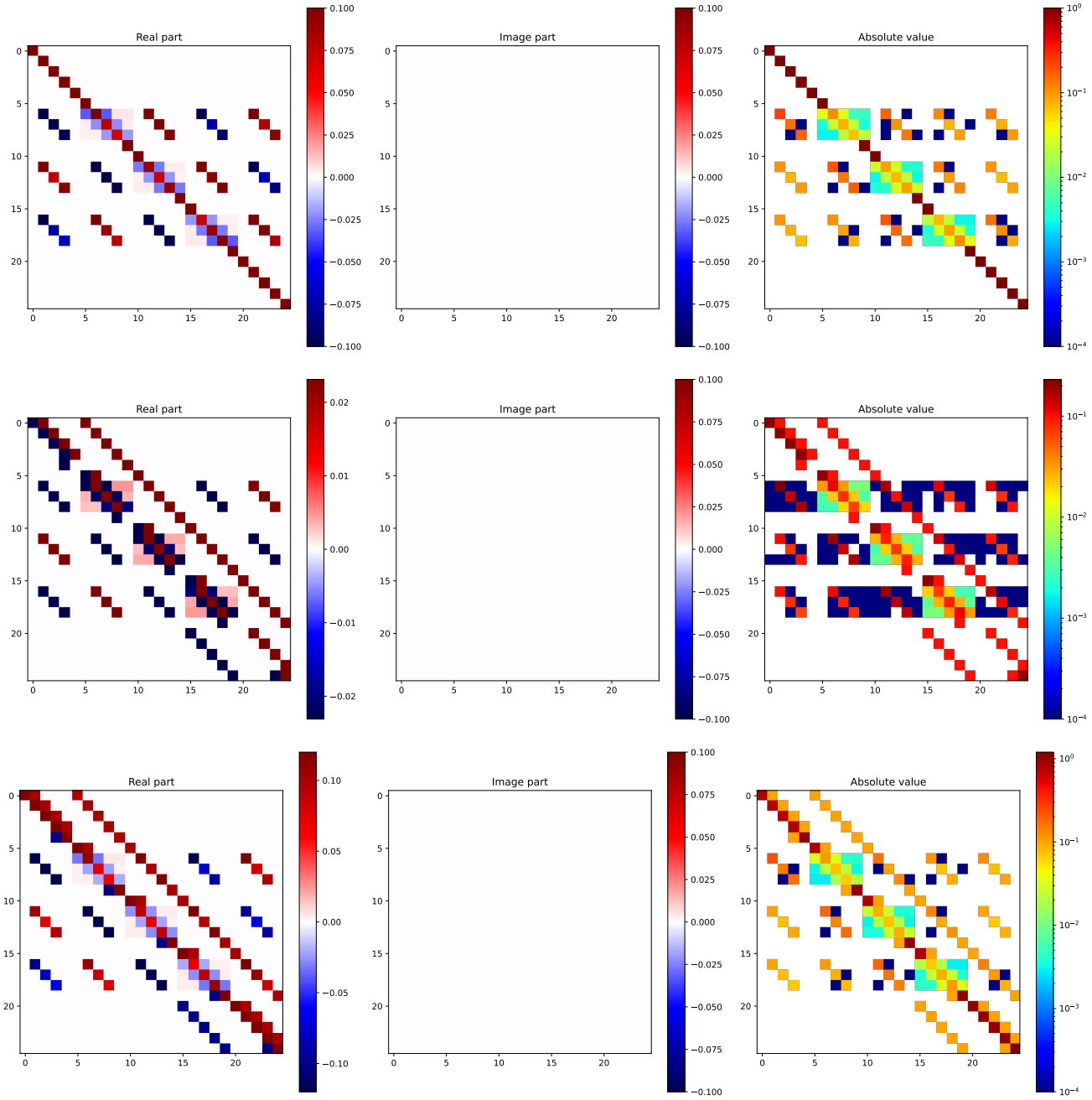
Visualization of three 25×25 FFT coefficient matrices



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