Neural operator preconditioning for accelerating the solution of the parametric Helmholtz equations

Yanfei Xiang, Luc Giraud, Paul Mycek, Maksym Shpakovych, Carola Kruse

Neural operator preconditioning for accelerating the solution of the parametric Helmholtz equations

SIAM LA24, Paris, France

May 15, 2024

Yanfei Xiang (✉ yanfei.xiang@inria.fr)

Concace team, Centre Inria de l'Université de Bordeaux

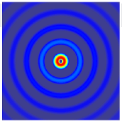
Parametric Helmholtz equations

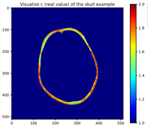
Helmholtz equation:

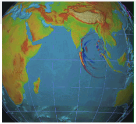
Parametric PDEs:
$$\begin{cases} \left[\nabla^2 + \left(\frac{\omega}{c(x)}\right)^2\right] u(x) = \rho(x) \\ \text{(subject to the Sommerfeld radiation condition at infinity)} \\ + \text{ Perfectly Matched Layer (PML) BCs on domain } \Omega \end{cases} \quad (1)$$

- let $x \in \mathbb{R}^d$ be the grid points in the d-dimensional Ω ($d = 1, 2, 3$);
- $u(x) : \mathbb{R}^d \to \mathbb{C}$ is the complex acoustic wavefield to be computed;
- $\rho(x) : \mathbb{R}^d \to \mathbb{C}$ is the source function (*could be fixed or not*);
- $c(x) : \mathbb{R}^d \to \mathbb{R}_+$ is a speed of sound distribution function (*could be fixed or not*) and $\omega = 1 \in \mathbb{R}_+$ is an angular frequency of the source.

Part of the Physical Applications:







Learning parametric Helmholtz operators

Transfer the differential expression to the discrete form:

$$\text{Parametric PDEs:} \begin{cases} \left[\nabla^2 + \left(\frac{\omega}{c(x)}\right)^2\right] u(x) = \rho(x) \\ + \text{ PML BCs on domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{ FEM, FDM, } \textbf{FFT}, \ldots \\ A(\mathrm{c})\mathrm{u} = \mathrm{b}, \text{ with LinearOp } A(\mathrm{c}) \in \mathbb{C}^{n \times n}, \ \mathrm{u}, \ \mathrm{b} \in \mathbb{C}^n \end{cases} \tag{2}$$

After discretization, numerical linear algebra methods, like subspace methods, can be used to solve the parametric Eq. (2). However,

- simply using subspace methods without preconditioning is much less effective for solving the Helmholtz Eqs.[a];
- 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 Helmholtz Eqs. (2).

[a] Ernst and Gander. Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods. 2012

Learning parametric Helmholtz operators

Transfer the differential expression to the discrete form:

$$\text{Parametric PDEs:} \begin{cases} \left[\nabla^2 + \left(\frac{\omega}{c(x)}\right)^2\right] u(x) = \rho(x) \\ + \text{ PML BCs on domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{ FEM, FDM, } \textbf{FFT}, \ldots \\ A(c)\mathrm{u} = \mathrm{b}, \text{ with LinearOp } A(c) \in \mathbb{C}^{n\times n}, \ \mathrm{u}, \ \mathrm{b} \in \mathbb{C}^n \end{cases} \tag{2}$$

In recent decade, the thrived neural networks (NNs) solvers, like the physics-informed neural networks (PINNs)[a], is used to solve the parametric Eq. (2) 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

Learning parametric Helmholtz operators

Transfer the differential expression to the discrete form:

$$\text{Parametric PDEs:} \begin{cases} \left[\nabla^2 + \left(\frac{\omega}{c(x)}\right)^2\right] u(x) = \rho(x) \\ + \text{ PML BCs on domain } \Omega \\ \quad \text{Discrete derivatives} \Downarrow \text{ FEM, FDM, } \textbf{FFT}, \ldots \\ A(\mathrm{c})\mathrm{u} = \mathrm{b}, \text{ with LinearOp } A(\mathrm{c}) \in \mathbb{C}^{n\times n}, \ \mathrm{u}, \ \mathrm{b} \in \mathbb{C}^n \end{cases} \quad (2)$$

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

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

The learned $\mathcal{F}_\theta$ can be used as a flexible preconditioner for the subspace methods (like FGMRES) to accelerate the solution of parametric Eq. (2) with varying c, varying b and varying domain Ω but **without re-training**.

* **Loss-function:**

$$\min_\theta \frac{\|A(\mathrm{c})\mathrm{u}_\theta - \mathrm{b}\|_2^2}{\|\mathrm{b}\|_2^2} \quad (3)$$

The generalized Arnoldi relation

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

$$AZ_m = V_{m+1}\bar{H}_m, \tag{4}$$

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.

In our FGMRES case, we have

$$z_i = \mathcal{F}_\theta(v_i), \; i = 1, \dots m, \tag{5}$$

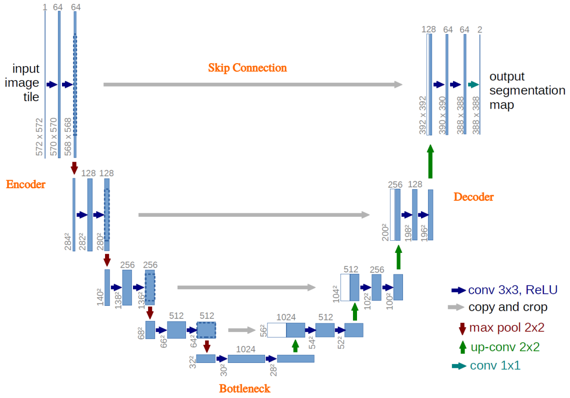
where $\mathcal{F}_\theta$ is the trained non-linear neural operator satisfies $\mathcal{F}_\theta \sim A^{-1}$. This part is computed with single machine precision (float 32), the one used in the training process, and other parts are in double precision.

$\Rightarrow$ 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

U-Net architecture

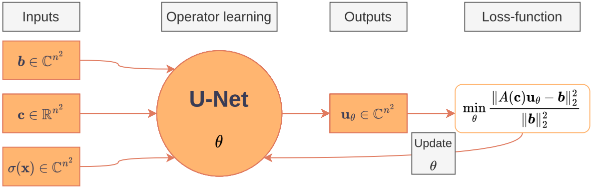
U-Net[a] architecture with 4 **depth**:



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][Figure 1] Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015

U-Net with meshes in 2D (domain 64×64)



Training (on 2 V100 GPUs, depth $= 4$, train. time: 49.83min):

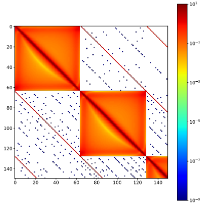
- Fixed grid point $\mathrm{x} \in \mathbb{R}^{\boldsymbol{n}^2}$ in the 2-dimensional domain ($n = 64$ for 2D case)
- Random source term $\mathrm{b} \in \mathbb{C}^{\boldsymbol{n}^2} \sim \mathcal{N}(0, 1)$ satisfying normally distr.
- Random speed of sound $\mathrm{c} \in \mathbb{R}^{\boldsymbol{n}^2}{}_+$ uniformly distributed on the interval [1, 2] and fixed frequency of the source $\omega = 1 \in \mathbb{R}_+$
- Function $\sigma(\mathrm{x}) : \mathbb{R}^{\boldsymbol{n}^2} \to \mathbb{C}^{\boldsymbol{n}^2}$ used in the definition of the PML boundary condition
- U-Net 2d: Training with single machine precision (float 32); Trainable params. - 832 K

Testing:

☺ Trained U-Net preconditioner can accelerate the solution of Eq. (2) with varying b c, domain size Ω (bcf. the discretisation invariance property from the convolution property)

Compare trained U-Net to algebraic preconditioner

Visualization of the first 150 sequential elements of a $64^2 \times 64^2$ matrix:

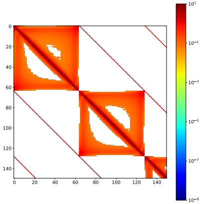


(a). Coefficient matrix A with $\frac{\text{nnz}}{n^2} \times \% = 33.38\%$

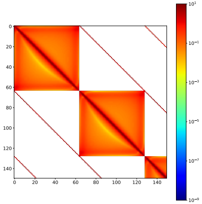
A with elements a_{ij} to be placed in matrix M.

$$(1).\ M(i,j) = \begin{cases} a_{ii}, & \text{if } \|a_{ii}\| \geq \texttt{thresh} \text{ for } i = j, \\ a_{ij}, & \text{if } \dfrac{\|a_{ij}\|}{\|a_{ii}\|\|a_{jj}\|} \geq \texttt{thresh} \text{ for } i \neq j. \end{cases}$$

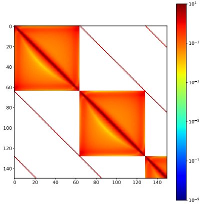
$$(2).\ \frac{\|A - M\|_F}{\|A\|_F} = \begin{cases} 5.41 \times 10^{-2} \text{ if } \texttt{thresh} = 10^{-3}, \\ 2.04 \times 10^{-3} \text{ if } \texttt{thresh} = 10^{-4}, \\ 1.81 \times 10^{-16} \text{ if } \texttt{thresh} = 10^{-5}. \end{cases}$$



(b). M with $\texttt{thresh}= 10^{-3}$, $\frac{\text{nnz}}{n^2} \times \% = 1.16\%$



(c). M with $\texttt{thresh}= 10^{-4}$, $\frac{\text{nnz}}{n^2} \times \% = 3.00\%$

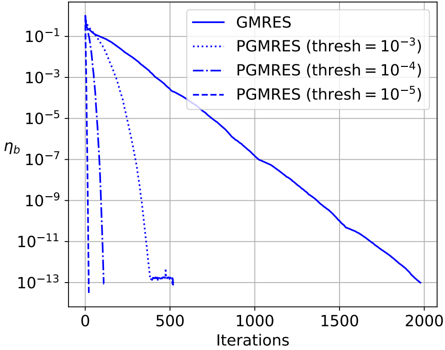


(d). M with $\texttt{thresh}= 10^{-5}$, $\frac{\text{nnz}}{n^2} \times \% = 3.10\%$

Compare trained U-Net to algebraic preconditioner

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

* All involved algorithms are run on the same CPUs/GPUs device with Python prototype.
* $\eta_b = \frac{\|A(c)u - b\|}{\|b\|}$. We stop the iteration when $\eta_b \leq \varepsilon$ with $\varepsilon = 10^{-13}$ by default.



PGMRES with varying `thresh`

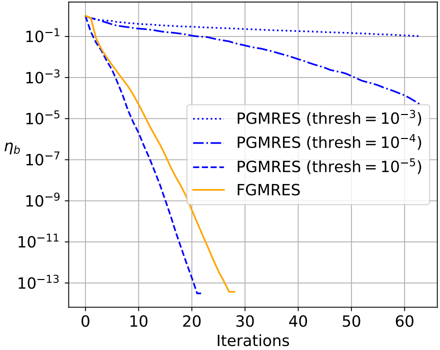
$$\frac{\|A - M\|_F}{\|A\|_F} = \begin{cases} 5.41 \times 10^{-2} & \text{if thresh} = 10^{-3}, \\ 2.04 \times 10^{-3} & \text{if thresh} = 10^{-4}, \\ 1.81 \times 10^{-16} & \text{if thresh} = 10^{-5}. \end{cases}$$

- PGMRES: GMRES preconditioned by `spilu`(M), which is realized by applying sparse `ilu` to the sparse matrix M with selected elements from coefficient matrix A

Compare trained U-Net to algebraic preconditioner

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

* FGMRES is GMRES preconditioned by the trained U-Net with depth = 4.
* FGMRES is implemented by the mixed-precision calculation.



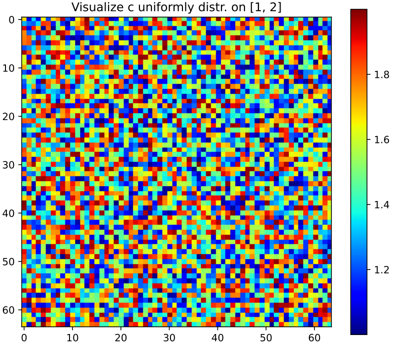
FGMRES and PGMRES

$$\frac{\|A - M\|_F}{\|A\|_F} = \begin{cases} 5.41 \times 10^{-2} \text{ if thresh} = 10^{-3}, \\ 2.04 \times 10^{-3} \text{ if thresh} = 10^{-4}, \\ 1.81 \times 10^{-16} \text{ if thresh} = 10^{-5}. \end{cases}$$

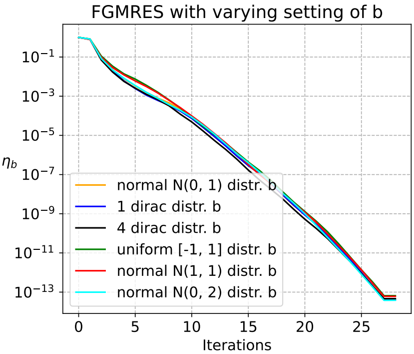
- The effectiveness of trained U-Net preconditioner is close to (could be the same as) spilu(M) with thresh= 10^{-5} (the best algebra preconditioner for this example)

Network generalizability: varying the source function b

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



(a). Visualize the fixed $\mathrm{c} \in \mathbb{R}^{64^2}$

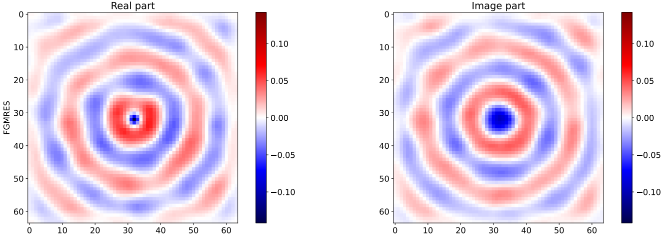


(b). Relative residual of FGMRES

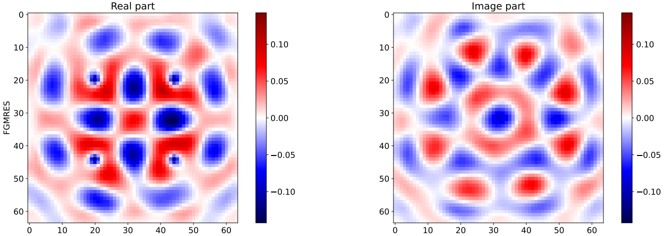
- Speed of sound $\mathrm{c} \in \mathbb{R}^{64^2}$ is fixed and satisfying uniform distribution on interval $[1, 2]$
- Performance of trained U-Net preconditioner is independent from varying the datatype of the $\mathrm{b} \in \mathbb{C}^{64^2}$ or varying the mean and median value of the normal setting of b

Network generalizability: varying the source function b

Visualize wavefiled $u \in \mathbb{C}^{64^2}$ solved by 1 dirac for $b \in \mathbb{C}^{64^2}$ (with fixed c):



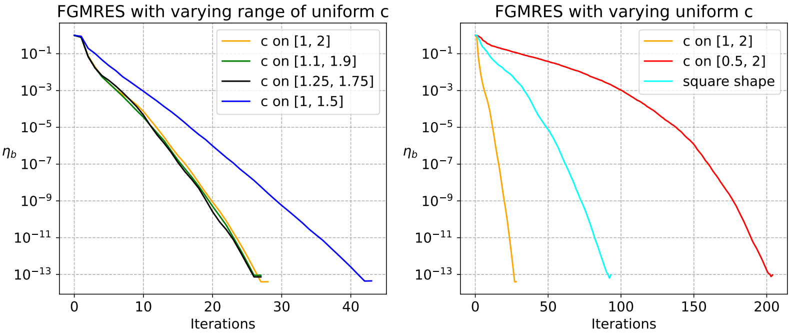
Visualize wavefiled $u \in \mathbb{C}^{64^2}$ solved by 4 dirac for $b \in \mathbb{C}^{64^2}$ (with fixed c):



Testing trained U-Net operator with $\mathcal{N}(0,1)$ source function on 2D

Network generalizability: varying the speed of sound c

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

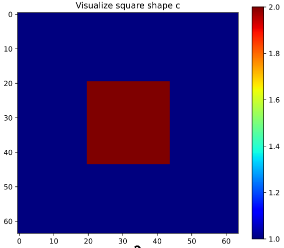


(a). shrink the interval (b). enlarge the interval & special structure

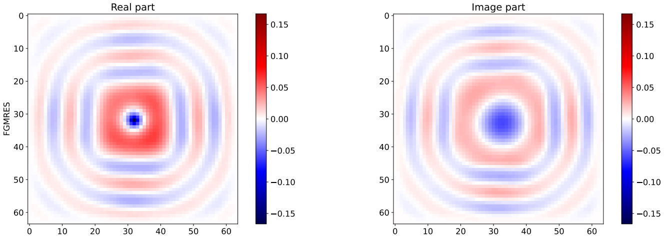
- Source function $\mathrm{b} \in \mathbb{C}^{64^2}$ is fixed with 1 dirac setting, 1 in grid position (32, 32), the center of domain 64×64, and 0 elsewhere in domain
- Square shape of $\mathrm{c} \in \mathbb{R}^{64^2}$: domain $[20:44, 20:44] = 2$ and 1 elsewhere
- Varying the range of the c satisfying uniform distribution and varying the its shape can effect the performance of the trained U-Net preconditioner

Network generalizability: varying the speed of sound c

Visualize $c \in \mathbb{R}^{64^2}$ with square shape:

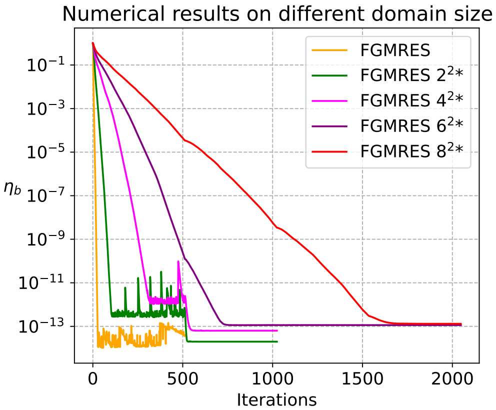


Visualize complex wavefiled $u \in \mathbb{C}^{64^2}$ solved by $c \in \mathbb{R}^{64^2}$ in square shape:



Network generalizability: varying the domain size

Solve $A(c)u = b$ from 2D domain 64×64 until 512×512 (i.e., from linear operator $A(c) \in \mathbb{C}^{64^2 \times 64^2}$ to $A(c) \in \mathbb{C}^{512^2 \times 512^2}$):



FGMRES — domain 64×64

FGMRES 2^2* — domain 128×128

FGMRES 4^2* — domain 256×256

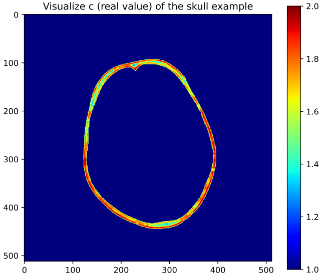
FGMRES 6^2* — domain 384×384

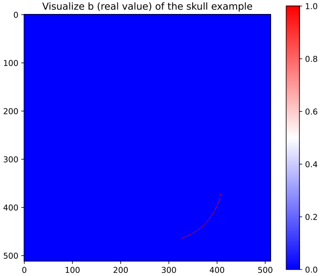
FGMRES 8^2* — domain 512×512

- U-Net trained on 2D domain 64×64 (with normal distr. $\mathcal{N}(0, 1)$ $b \in \mathbb{C}^{64^2}$ and uniform distr. $c \in \mathbb{R}^{64^2}$ in interval [1,2]) exhibits excellent network generalizability from the 4x larger domain 128×128 until the **64x larger** domain 512×512

Network generalizability: Adult human skull example

Solve $A(\mathrm{c})\mathrm{u} = \mathrm{b}$ with the **structured dataset** for both $\mathrm{c} \in \mathbb{R}^{512^2}$ and $\mathrm{b} \in \mathbb{R}^{512^2}$ on a 64x larger 2D domain 512×512 ($A(\mathrm{c}) \in \mathbb{C}^{512^2 \times 512^2}$):





This example is downloaded from the qure.ai dataset[a]

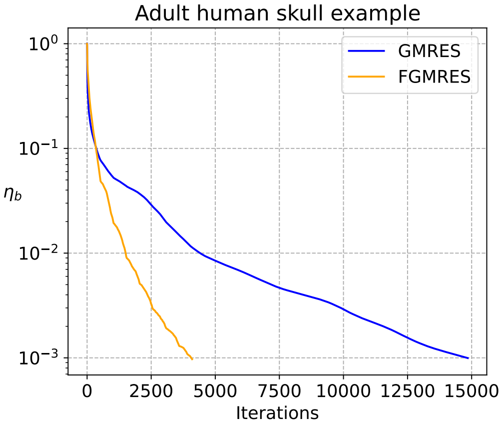
Recall part of the training settings:

- Random training datasets on domain 64×64: $\mathrm{b} \in \mathbb{C}^{64^2}$ satisfies standard Normal distr. $\mathcal{N}(0, 1)$; $\mathrm{c} \in \mathbb{R}^{64^2}$ is Uniform distr. on interval $[1, 2]$

[a] http://headctstudy.qure.ai/dataset

Network generalizability: Adult human skull example

Solve $A(c)u = b$ with the **structured dataset** for both c and b on 2D domain 512×512 (i.e., $A(c) \in \mathbb{C}^{512^2 \times 512^2}$):

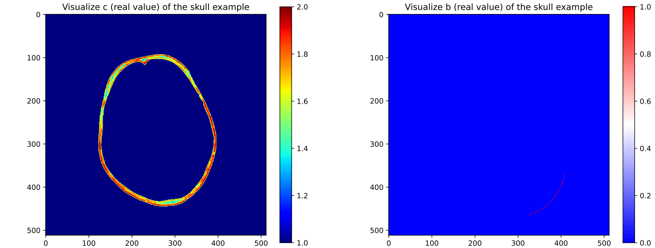


- Comparison of FGMRES (NNs trained on domain 64×64) to GMRES on solving the skull example (on 512×512 from CQ500 with 491 scans) to reaching accuracy at 10^{-3}
- Consumed its & CPU time (Training time of U-Net on 2 V100 GPUs is 49.83min): FGMRES = 4097 & 4.24h; GMRES = 14849 & 13.16h

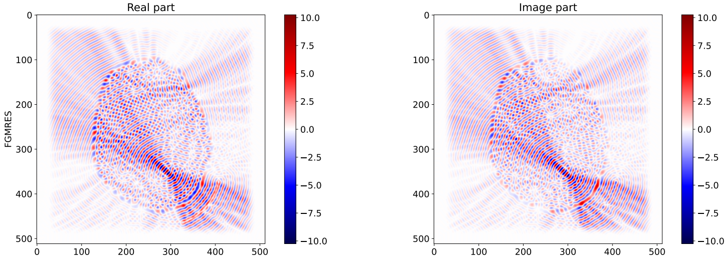
Testing trained U-Net operator with $\mathcal{N}(0,1)$ source function on 2D

Network generalizability: Adult human skull example

Visualization the structured dataset of the skull example on 2D domain 512×512:



Visualize the complex wavefiled $c \in \mathbb{R}^{512^2}$ of human skull example solved by FGMRES:



PINNs & DeepONet & FNO

Other neural networks (NNs) architectures for the parametric Helmholtz equations:

- NNs solver:
  - PINNs: Lu et al., Physics-Informed Neural Networks with Hard Constraints for Inverse Design. SISC. 2021
- NNs operators:
  - DeepONet: Lu et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. 2021;
  - FNO: Kovachki et al., Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. 2023

$\Rightarrow$ Compare to above NNs architectures, **U-Net** is the optimal one that meets all our goal and exhibits the robust preconditioning behavior

Three random setting for the source function b:

- **Normal distr.** $\mathcal{N}$**(0,1)**: satisfying the standard normal distribution $\mathcal{N}(0,1)$;
- Uniform distr. $[-1, 1]$: satisfying the uniformly distribution on interval $[-1, 1]$;
- Dirac setting: with one non-zero value in the random position in the 2D domain;

Take home message of this work

Goal: To learn the optimal neural operator $\mathcal{F}_\theta$ for accelerating the solution of the parametric Helmholtz Eqs.

Training different neural networks (NNs) $\Rightarrow$ choose the optimal NNs architecture —> **U-Net is the optimal one for our case**

Training U-Net on 2D with different settings for b $\Rightarrow$ choose the optimal training dataset —> **Standard Normal distr. $\mathcal{N}$(0,1) for** b

$\Rightarrow$ The performance of trained neural operator depends on:

- the choosing of NNs architecture;
- the setting of training datasets;
- the tuning of hyper-parameters of NNs;
- the utilities of training tools like pytorch-lightning.

⌚ Research report version of this work will soon be accessible online at HAL (https://inria.hal.science/) !

$\longrightarrow$ Yanfei Xiang (yanfei.xiang@inria.fr)

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.*

Acknowledgement

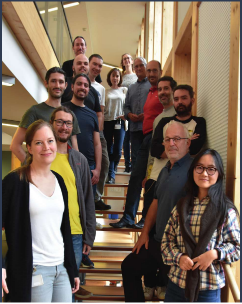
Concace

Numerical & Parallel Composability for High Performance Computing

Joint Inria-Industry Project Team with Airbus Central R&T and Cerfacs

To deal with large size and large dimension problems coming from model- and data-driven applications, Concace will take advantage of modern development tools and languages to design high-level expressions of complex parallel algorithms.

While the traditional approach to HPC is to fully exploit hardware, Concace's complementary approach will enable a richer composability of numerical methods, allowing to fully exploit existing and new numerical algorithms.

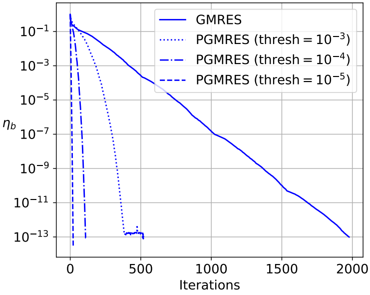


Thank you for your attention!

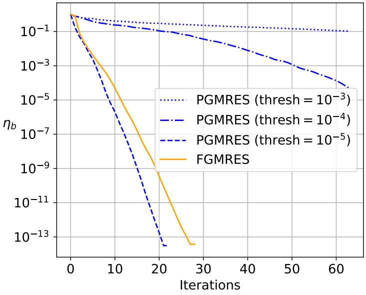
Questions ?

Consuming time

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



(a). PGMRES with varying `thresh`



(b). FGMRES and PGMRES

- PGMRES: GMRES precondioned by `spilu`(M), which is realized by applying sparse `ilu` to the sparse matrix M with selected elements from coefficient matrix A
- The effectiveness of trained U-Net preconditioner is close to (could be the same as) `spilu`(M) with `thresh`$= 10^{-5}$ (the best algebra preconditioner for this example)
- Iterations & **GPUs time**: GMRES: 1978 & 141.8737s;
  FGMRES: 29 & 4.3403s; PGMRES (`thresh`$= 10^{-3}$): 522 & 43.0996s;
  PGMRES (`thresh`$= 10^{-4}$): 111 & 3.2276s; PGMRES (`thresh`$= 10^{-5}$): 23 & 0.6582s