# Hybridization of Machine Learning and Numerical Linear Algebra Techniques for Scientific Computing: Learned Minimum Residual Solvers for the Helmholtz Equations

Yanfei Xiang, Luc Giraud, Paul Mycek, Carola Kruse

# Hybridization of machine learning and numerical linear algebra techniques for scientific computing

SIAM CSE23, MS110, February 28, 2023

**Yanfei Xiang** (yanfei.xiang@inria.fr - Alpines team)
Concace team, Centre Inria de l'université de Bordeaux
Joint work with: Luc Giraud, Paul Mycek and Carola Kruse

# Summary





Inria

# 1 Scientific machine learning

Research on Machine Learning (ML), especially deep learning with Deep Neural Networks (DNN), has been increasingly applied to scientific computing (called **Scientific Machine Learning - SciML**), particularly for problems related to solve the Partial Differential Equations (PDE).

**Three main directions** of this SciML trend:

1. ML algorithms for devising **a recommendation system** to assist the optimal-selection of traditional methods (auto-selecting of the best solvers/preconditioner/restart parameter etc.), such as the SALSA (2006) and Lighthouse (2016) projects (Sood, 2019);

Research on Machine Learning (ML), especially deep learning with Deep Neural Networks (DNN), has been increasingly applied to scientific computing (called **Scientific Machine Learning - SciML**), particularly for problems related to solve the Partial Differential Equations (PDE).

**Three main directions** of this SciML trend:

1. ML algorithms for devising **a recommendation system** to assist the optimal-selection of traditional methods (auto-selecting of the best solvers/preconditioner/restart parameter etc.), such as the SALSA (2006) and Lighthouse (2016) projects (Sood, 2019);
2. Use data-driven DNN to **build a solver directly** for the simulations of PDE, such as the Physics-Informed Neural Network (PINN) (Lu et al., 2020-2023, and others), pure DNN solver for CFD problem (Tompson et al., 2017);

Research on Machine Learning (ML), especially deep learning with Deep Neural Networks (DNN), has been increasingly applied to scientific computing (called **Scientific Machine Learning - SciML**), particularly for problems related to solve the Partial Differential Equations (PDE).

**Three main directions** of this SciML trend:

1. ML algorithms for devising **a recommendation system** to assist the optimal-selection of traditional methods (auto-selecting of the best solvers/preconditioner/restart parameter etc.), such as the SALSA (2006) and Lighthouse (2016) projects (Sood, 2019);
2. Use data-driven DNN to **build a solver directly** for the simulations of PDE, such as the Physics-Informed Neural Network (PINN) (Lu et al., 2020-2023, and others), pure DNN solver for CFD problem (Tompson et al., 2017);

- **Challenges** (Stability, Accuracy, Computational cost, and Curse of dimensionality, other black-boxes etc.), and **mathematical interpretation** of SciML methods based on DNN (Adcock and Dexter's recent work, 2020-2022);

Research on Machine Learning (ML), especially deep learning with Deep Neural Networks (DNN), has been increasingly applied to scientific computing (called **Scientific Machine Learning - SciML**), particularly for problems related to solve the Partial Differential Equations (PDE).

**Three main directions** of this SciML trend:

1. ML algorithms for devising **a recommendation system** to assist the optimal-selection of traditional methods (auto-selecting of the best solvers/preconditioner/restart parameter etc.), such as the SALSA (2006) and Lighthouse (2016) projects (Sood, 2019);
2. Use data-driven DNN to **build a solver directly** for the simulations of PDE, such as the Physics-Informed Neural Network (PINN) (Lu et al., 2020-2023, and others), pure DNN solver for CFD problem (Tompson et al., 2017);
   - **Challenges** (Stability, Accuracy, Computational cost, and Curse of dimensionality, other black-boxes etc.), and **mathematical interpretation** of SciML methods based on DNN (Adcock and Dexter's recent work, 2020-2022);
3. **Hybrid** SciML and the traditional methods (Rizzuti et al., 2019, Illarramendi et al., 2020) ← **Our interests**.

A 2D Helmholtz equation with a heterogeneous sound speed distribution (sequences of linear systems with multiple left hand sides) is described as

$$A^{(\ell)}x^{(\ell)} = b,\ \ell = 1, 2, \ldots \text{(family index)}, \tag{1}$$

where $A^{(\ell)}$ are slowly-varying complex sparse matrices, $x^{(\ell)}$ is the complex solution to be approximated (omit $^{(\ell)}$ later), and $b$ is the fixed right-hand side.

A 2D Helmholtz equation with a heterogeneous sound speed distribution (sequences of linear systems with multiple left hand sides) is described as

$$A^{(\ell)}x^{(\ell)} = b,\ \ell = 1, 2, \ldots \text{(family index)}, \tag{1}$$

where $A^{(\ell)}$ are slowly-varying complex sparse matrices, $x^{(\ell)}$ is the complex solution to be approximated (omit $^{(\ell)}$ later), and $b$ is the fixed right-hand side.

Candidate iterative methods (the traditional ones & the recently SciML):

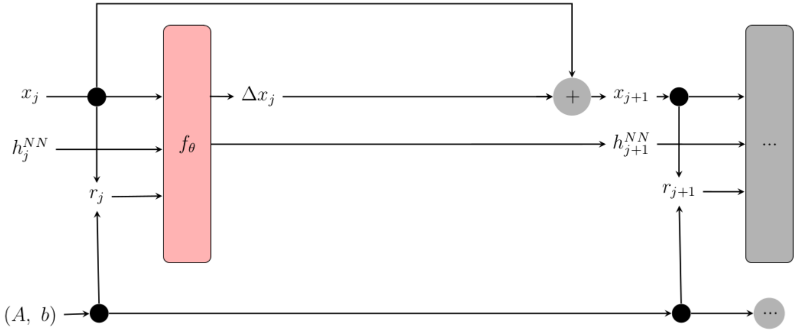
— Krylov subspace methods: like GMRES (Saad book, 2003).

— A recurrent Neural Network (NN) solver with non-linear fixed-point iterative scheme (Stanziola et al., JCP, 2021) ($x_j$: approximated $x$ at the $j$-iteration):

$$\begin{aligned} r_j &= b - Ax_j, \\ \Delta x_j &= f_\theta(x_j, r_j), \\ x_{j+1} &= x_j + \Delta x_j. \end{aligned}$$

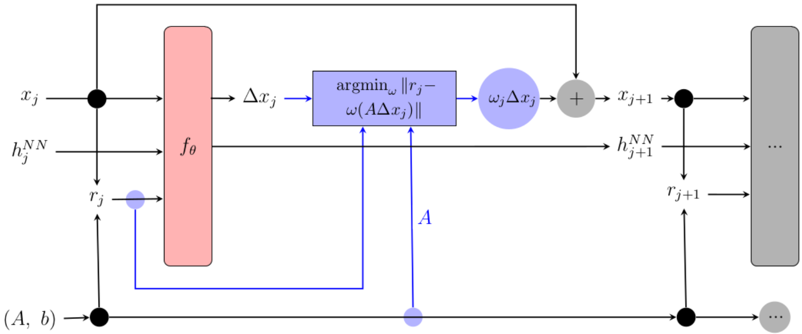
- $f_\theta$: NN with a modified U-Net architecture (Ronneberger et al., 2015)
  - Loss function of NN: a physics-based loss function embed the residual $r_j$

# Optimal step size $\omega_j$



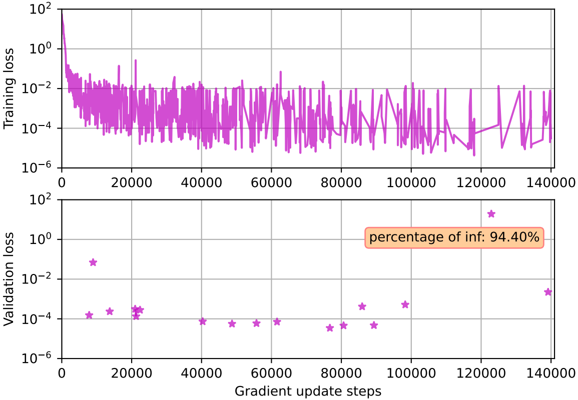

The $f_\theta$ is the NN with a modified U-Net architecture and a physics-based loss function embed the mean squared error (MSE) of the linear system residual $r_j$

- Solution-update architecture of R(R-NN): $x_{j+1} = x_j + \Delta x_j$
- Term R denotes Richardson-like iteration scheme, and R-NN stands for corresponding NN-inference



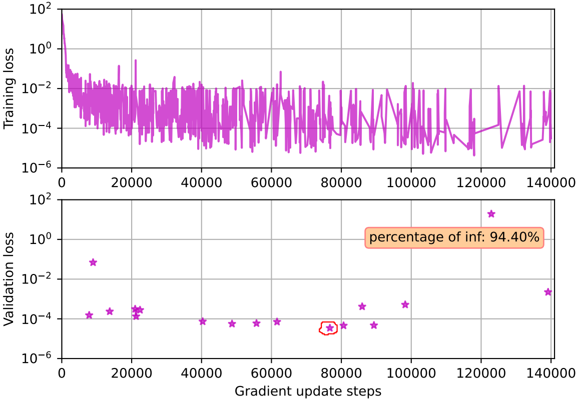

The $f_\theta$ is the NN with a modified U-Net architecture and a physics-based loss function embed the mean squared error (MSE) of the linear system residual $r_j$

- Solution of MRR(MRR-NN): $x_{j+1} = x_j + \omega_j \Delta x_j$ with $\omega_j = (A\Delta x_j)^H r_j / \|A\Delta x_j\|_2^2$
- Term MRR denotes Minimum Residual Richardson iteration scheme, and MRR-NN stands for corresponding NN-inference



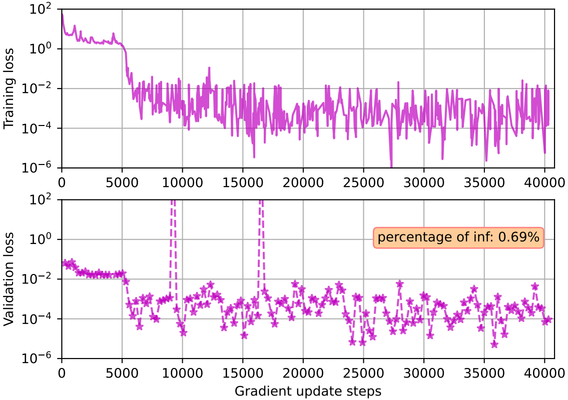

R(R-NN):

- Many infinite values exist in the validation loss (poor robustness)



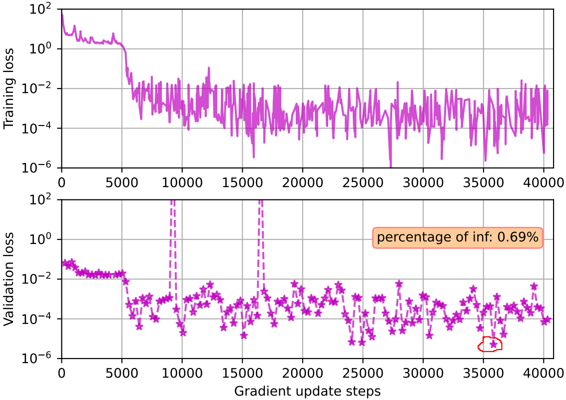

R(R-NN):

- Many infinite values exist in the validation loss (poor robustness)
- Reach the smallest validation loss with value slightly lower than $10^{-4}$ around step 80000




MRR(MRR-NN):

- Show more robustness with much less infinite values in the validation loss




MRR(MRR-NN):

- Show more robustness with much less infinite values in the validation loss
- Reach the smallest validation loss with value lower than $10^{-5}$ around step 35000

The **generalized Arnoldi relation** when preconditioner is applied in the Krylov subspace methods

$$AZ_j = V_{j+1}\underline{H}_j,\ Z_j = [z_1, \cdots, z_j]. \tag{2}$$

FGMRES (Flexible GMRES) with varying preconditioner:

$$x_j = x_0 + Z_j y_j,\ y_j = \mathrm{argmin}_y \left\| \beta e_1 - \underline{H}_j y \right\| \text{ (Minimal residual)}$$

The **generalized Arnoldi relation** when preconditioner is applied in the Krylov subspace methods

$$AZ_j = V_{j+1}\underline{H}_j,\ Z_j = [z_1, \cdots, z_j]. \qquad (2)$$

FGMRES (Flexible GMRES) with varying preconditioner:

$$x_j = x_0 + Z_j y_j,\ y_j = \mathrm{argmin}_y \left\| \beta e_1 - \underline{H}_j y \right\| \text{ (Minimal residual)}$$

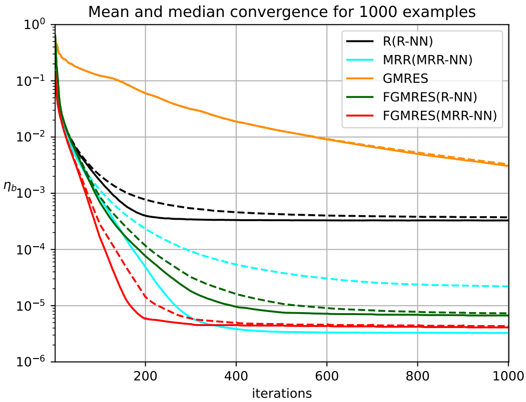
**Strategies for using trained neural network $f_\theta$ as preconditioner**

- Strategy 1: "**Krylov driven**" $z_j \approx A^{-1} v_j$, compute $z_j = f_\theta(0, v_j)$
- Strategy 2: "**NN driven**", compute $z_j = f_\theta(x_{j-1}, r_{j-1}/\|r_{j-1}\|)$

**Subspace methods with trained neural network preconditioner**

- FGMRES(MRR-NN): Flexible GMRES method preconditioned by the trained MRR-NN inference
- Only Strategy 2 for FGMRES(*) is presented in the rest of slides

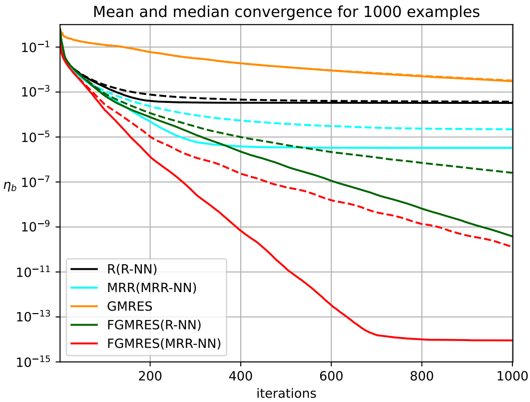
$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$




fp32/32-bit calculation (dashed line – mean; solid line – median):

- Better attainable accuracy of the MRR(MRR-NN), and the preconditioned methods (FGMRES(R-NN) and FGMRES(MRR-NN) with Strategy 2)
- Plateaus in NN-solvers and preconditioned ones are caused by different reasons

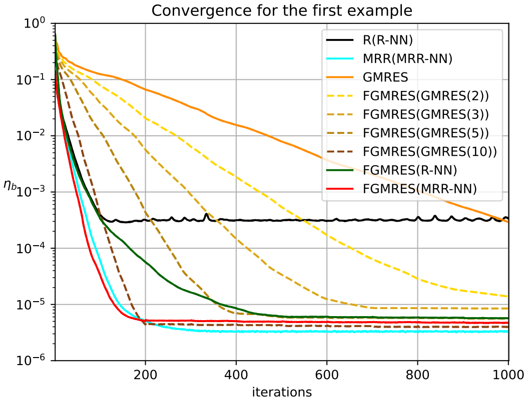
$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$




mixed arithmetic calculation — fp32 & fp64 (fp32 only for the NN part):

- Plateaus of FGMRES(R-NN) and FGMRES(MRR-NN) are removed (except one of FGMRES(MRR-NN) that is restricted by fp64 precision)
- No change in first three solvers because they already reach the best they could

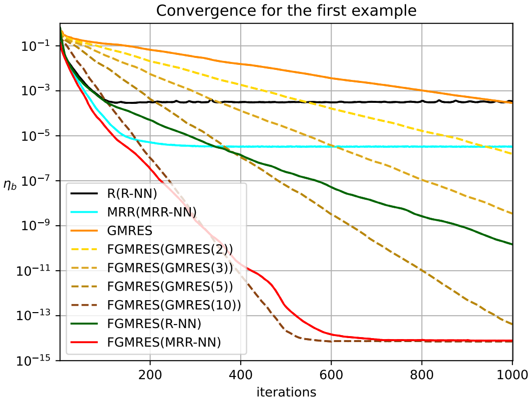
$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$




FGMRES with NN and GMRES($m$) preconditioner (fp32):

- Preconditioned variant FGMRES(*-NN) with NNs still performs the best (in terms of the final attainable accuracy $\eta_b$ and the speed to reach the plateau)
- Increasing the value of $m$ in FGMRES(GMRES($m$)) can improve its performance

$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$




FGMRES with NN and GMRES($m$) preconditioner (fp32 & fp64):

- Preconditioned variant FGMRES(*-NN) with NN still performs the best
- Increasing the value of $m$ in FGMRES(GMRES($m$)) can improve its performance
- Plateau of the preconditioned variants is restricted by the working precision

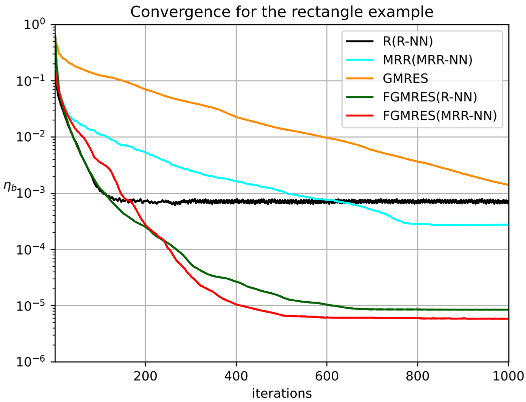
| # example | Method | $\eta_b$(fp32 / fp32&fp64) | $\#time(s)$ |
|---|---|---|---|
| 1st | R(R-NN) | 3.36e-04 / 3.36e-04 | 7.66 / 10.48 |
| | MRR(MRR-NN) | 3.27e-06 / 3.27e-06 | 9.52 / 14.07 |
| | GMRES | 2.92e-04 / 2.88e-04 | 18.00 / 31.78 |
| | FGMRES(GMRES(2)) | 1.39e-05 / 1.57e-06 | 27.42 / 54.92 |
| | FGMRES(GMRES(3)) | 8.50e-06 / 3.48e-09 | 31.67 / 52.15 |
| | FGMRES(GMRES(5)) | 5.64e-06 / 4.19e-14 | 42.92 / 53.68 |
| | FGMRES(GMRES(10)) | 3.99e-06 / 7.17e-15 | 79.13 / 95.33 |
| | FGMRES(R-NN) | 5.69e-06 / 1.48e-10 | 24.58 / 30.39 |
| | FGMRES(MRR-NN) | 4.72e-06 / 7.82e-15 | 24.23 / 31.04 |

TABLE 1

- FGMRES(GMRES($m$)) could be as competitive as FGMRES(*-NN) if the value of $m$ is large. However, this also increases the implementation time
- Even FGMRES(GMRES(2)) requires more implementation time to reach a worse attainable accuracy compared to FGMRES(R-NN)/FGMRES(MRR-NN)
- Observation in reaching better attainable accuracy with less implementation time verifies the advantages of the NNs precondtioner

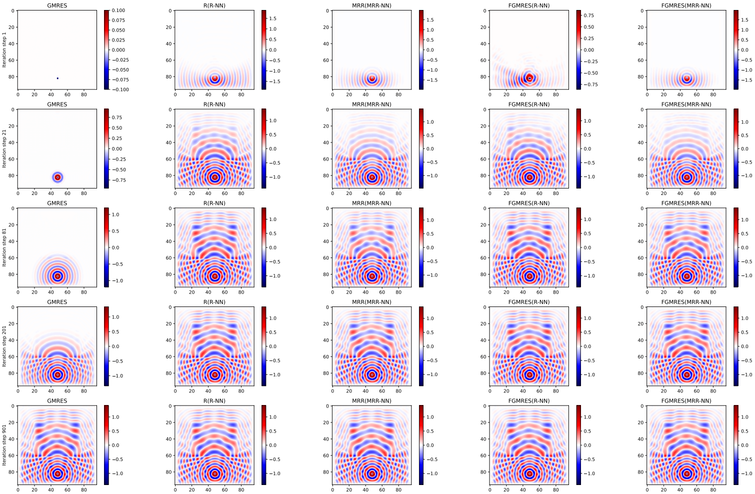
# Network generalizability I: Rectangular shape

$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$

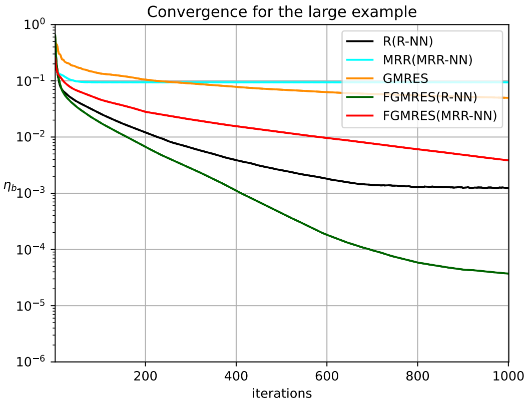



change geometric from circular or elliptic shape into rectangular shape (fp32):

- Better attainable accuracy of MRR(MRR-NN) but with more iterations
- Apply the trained NN-inferences as a preconditioner still works



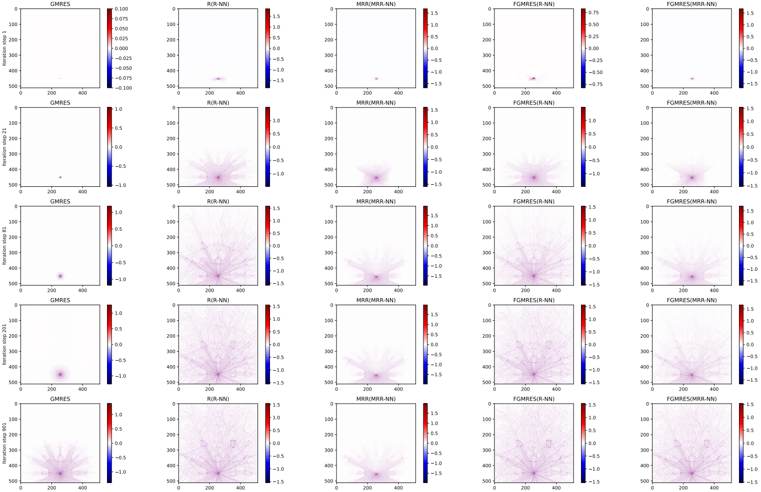

- Rectangle example: Rectangular shape with a background sound speed of 2 m/s




$$\eta_b = \frac{\|Ax_j - b\|_2}{\|b\|_2}$$

from domain on $96 \times 96$ grid points to large domain on $480 \times 480$ (fp32):

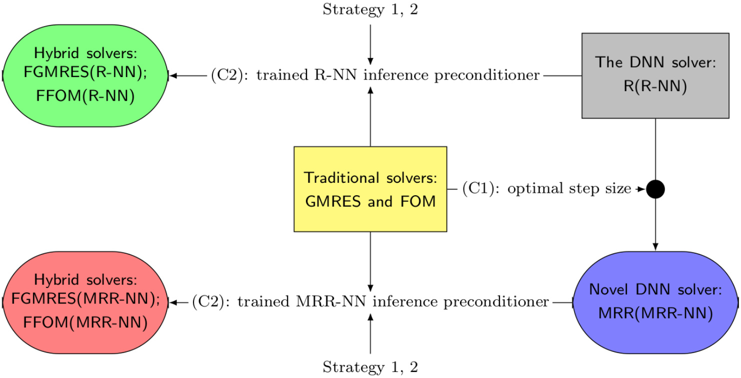
- For some unknown reasons, MRR(MRR-NN) stagnates at the early iterations in lower attainable accuracy, and thus sub-performances than R(R-NN)
- Apply the trained NN-inferences as a preconditioner still works




- Large example: expand to large domain on 480 × 480 grid points

# 2 Conclusions & Perspectives

Hybridization of subspace methods and machine learning:




- Two main contributions (simplified as C1 and C2) in hybridizing machine learning and subspace methods (DNN refers to deep neural networks)

**For the scientific machine learning methods**:

- Explore the balance between the **better attainable accuracy** and the **good generalizability** of network;
- Address the **vanishing gradient issue** (especially when the loss function is related to the PDE residual with a better attainable accuracy);
- Devise the **loss function** with more information;

**For the scientific machine learning methods**:

- Explore the balance between the **better attainable accuracy** and the **good generalizability** of network;
- Address the **vanishing gradient issue** (especially when the loss function is related to the PDE residual with a better attainable accuracy);
- Devise the **loss function** with more information;

- Choose **other neural network architectures**;
- Try **other hyper-parameters** (like optimizer, batch size, learning rate, etc.) and study their effects in performance.

***Scientific machine learning part (Chapter 5) of my PhD thesis.***

*Registration and travel support for this presentation was provided by the Society for Industrial and Applied Mathematics & Concace team, Centre Inria de l'université de Bordeaux*

*Thank you Prof. Eric de Sturler for the invitation!*

*Scientific machine learning part (Chapter 5) of my PhD thesis.*

*Thank you for your attention!*

# Back slides: Architecture of modified U-Net

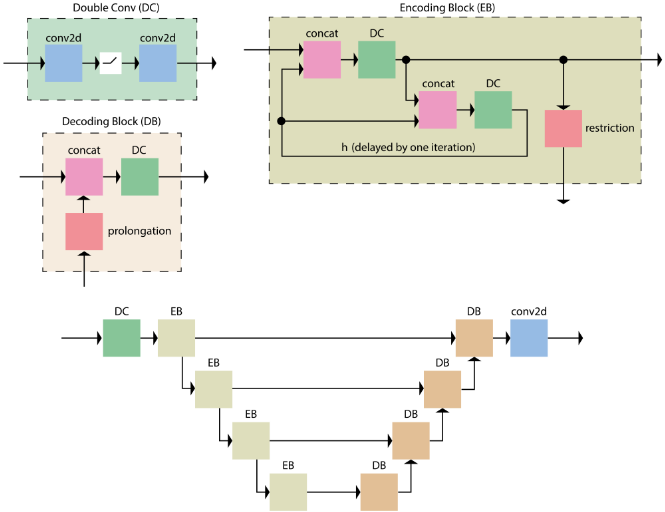
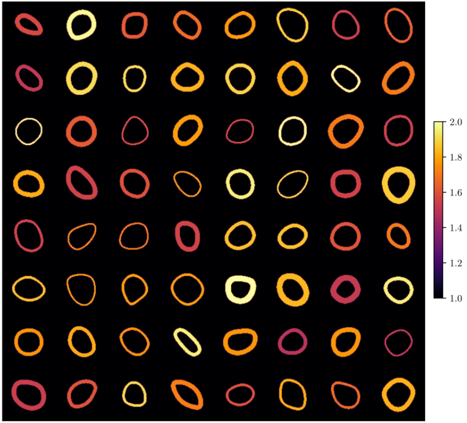
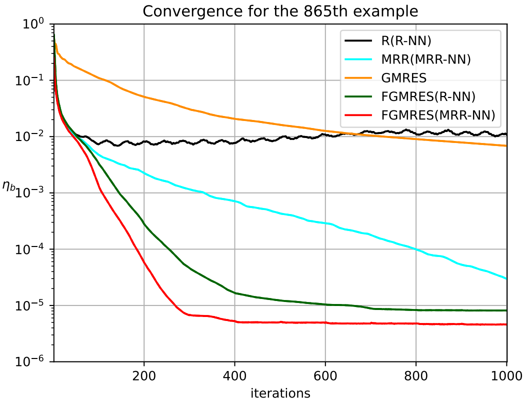



**Fig. 3.** Architecture of the modified UNet used for the learned optimizer $f_\theta$. Each encoding block (EB) contains two double convolution (DC) layers, one to compute the output passed to subsequent layers, and one to compute the hidden state $h$. The concat blocks stack the inputs in the channel dimension. The network is lightweight, with only 8 channels per convolutional block at every scale and a total of 47k trainable parameters.

# Back slides: The idealized skulls

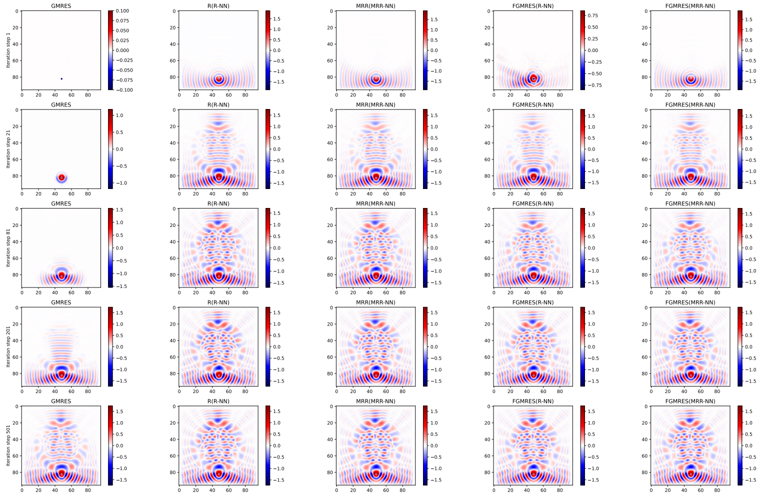



- Part examples of the heterogeneous sound speed distributions based on idealized skulls used to train the two NN solvers. Each skull is created by summing up several circular harmonics of random amplitude and phase, and then assigned a random thickness between 2 and 10 pixels, and a random sound speed between 1.5 and 2 times the background value. (Fig.4. of Stanziola et al.,'s JCP, 2021)



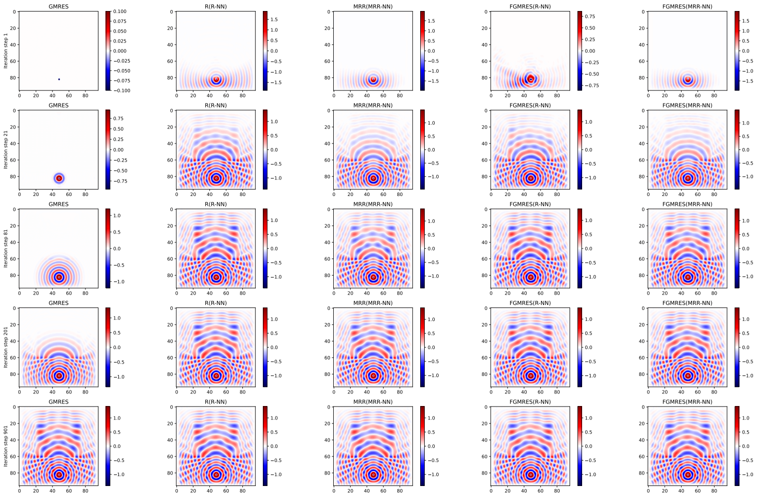

test the 865th example with domain on $96 \times 96$ grid points (fp32):

- For some unknown reasons, MRR(MRR-NN) stagnates at the early iterations in lower attainable accuracy, and thus sub-performances than R(R-NN)
- Apply the trained NN-inferences as a preconditioner still works

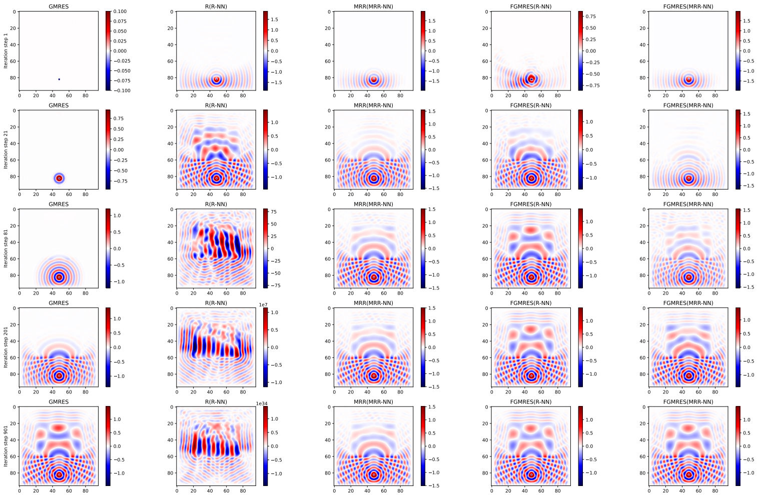



- The 865th example selected from test data set with idealized skulls shapes and a background sound speed of 1 m/s on a bounded domain $96 \times 96$ grid points

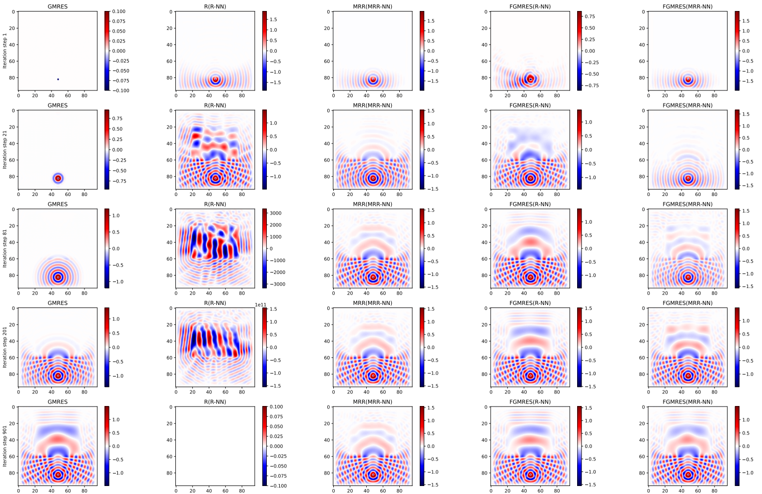
# Back slides: generalizability I with more exps



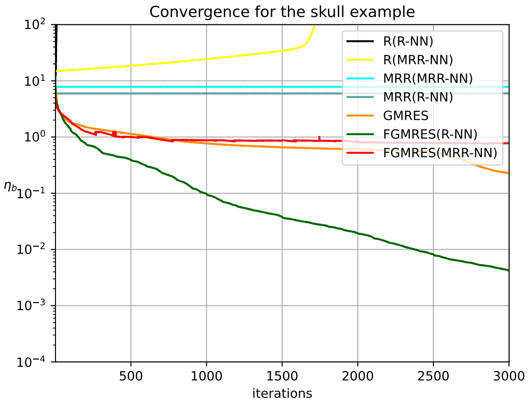

- Rectangle example: Rectangular shape with a background sound speed of 2 m/s




- Rectangle example: Rectangular shape with a background sound speed of **3 m/s**

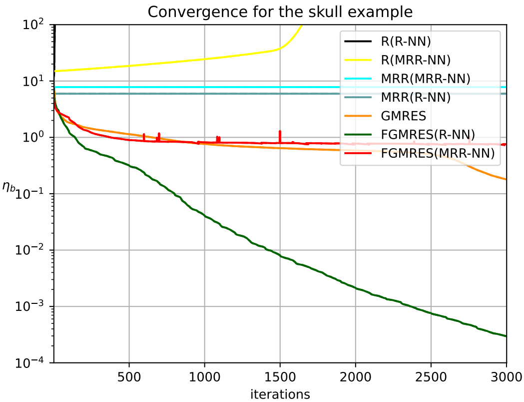



- Rectangle example: Rectangular shape with a background sound speed of **4 m/s**



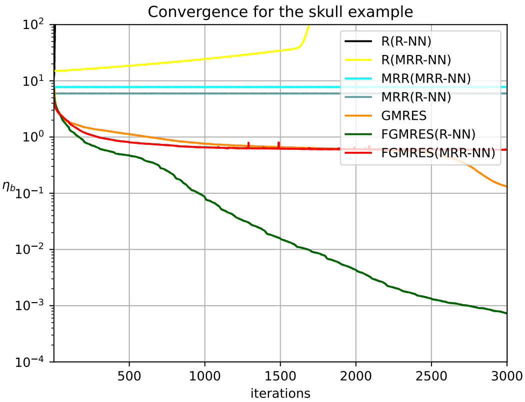

CT000100 (fp32):

- For some unknown reasons, MRR(MRR-NN) stagnates and R(R-NN) fails to convergence
- Apply the trained NN-inferences as a preconditioner still works




CT000200 (fp32):

- For some unknown reasons, MRR(MRR-NN) stagnates and R(R-NN) fails to convergence
- Apply the trained NN-inferences as a preconditioner still works




CT000225 (fp32):

- For some unknown reasons, MRR(MRR-NN) stagnates and R(R-NN) fails to convergence
- Apply the trained NN-inferences as a preconditioner still works

# Back slides: Bibliography I


1 [GCRO] E. de Sturler. Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics*, 67(1):15–41, 1996, DOI: 10.1016/0377-0427(94)00123-5

2 [D-CG] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'H. A Deflated Version of the Conjugate Gradient Algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000, DOI: 10.1137/S1064829598339761

3 [GMRES-DR] R. B. Morgan. GMRES with Deflated Restarting. *SIAM Journal on Scientific Computing*, 24(1):20-37, 2002, DOI: 10.1137/S1064827599364659

4 [GCRO-DR] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov Subspaces for Sequences of Linear Systems. *SIAM Journal on Scientific Computing*, 28(5):1651-1674, 2006, DOI: 10.1137/040607277

5 [GCRO-DR] L. M. Carvalho, S. Gratton, R. Lago, and X. Vasseur. A Flexible Generalized Conjugate Residual Method with Inner Orthogonalization and Deflated Restarting. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1212-1235, 2011, DOI: 10.1137/100786253

6 [Survey] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer. A survey of subspace recycling iterative methods. *Special Issue: Topical Issue Applied and Numerical Linear Algebra - Part II*, 43(4), 2020, DOI: 10.1002/gamm.202000016

7 [BCG] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980, DOI: 10.1016/0024-3795(80)90247-5

8 [BGMRES, Saad book] Y. Saad. Iterative Methods for Sparse Linear Systems, 2nd ed. SIAM, Philadelphia, 2003.

9 [IB-BGMRES] M. Robbé and M. Sadkane. Exact and inexact breakdowns in the block GMRES method. *Linear Algebra and its Applications*, 2006, 419:(1), pp. 265-285, DOI: 10.1016/j.laa.2006.04.018

10 [BFOM] A. Frommer, K. Lund, and D. B. Szyld. Block Krylov Subspace Methods for Functions of Matrices II: Modified Block *SIAM Journal on Matrix Analysis and Applications*, 41(2):804–837, 2020, DOI: 10.1137/19M1255847

11 [Block Krylov&Stability] E. Carson, K. Lund, M. Rozloznik, and S. Thomas. Block Gram-Schmidt algorithms and their stability properties. *Linear Algebra and its Applications*, 638:150–195, 2022, DOI: 10.1016/j.laa.2021.12.017

12 [BGMRES-DR] R. B. Morgan. Restarted block-GMRES with deflation of eigenvalues. *Applied Numerical Mathematics*, 2005, 54:(2), pp. 222-236, DOI: 10.1016/j.apnum.2004.09.028

13 [IB-BGMRES-DR] E. Agullo, L. Giraud and Y.-F. Jing. Block GMRES Method with Inexact Breakdowns and Deflated Restarting. *SIAM Journal on Matrix Analysis and Applications*, 2014, 35:(4), pp. 1625-1651, DOI: 10.1137/140961912

14 [BGCRO-DR] M. L. Parks, K. M. Soodhalter and D. B. Szyld. A block Recycled GMRES method with investigations into aspects of solver performance. *arXiv*, 2016, abs/1604.01713

15 [D-SBGMRES-DR] A. Tajaddini, G. Wu, F. Saberi-Movahed, and N. Azizizadeh. Two New Variants of the Simpler Block GMRES Method with Vector Deflation and Eigenvalue Deflation for Multiple Linear Systems. *Journal of Scientific Computing*, 86:9, 2021, DOI: 10.1007/s10915-020-01376-w

16 [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

17 [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

18 [Helmholtz equation] A. Stanziola, S. R. Arridge, B. T. Cox, and B. E. Treeby. A Helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound. *Journal of Computational Physics*, 441(2):110430, 2021, DOI: 10.1016/j.jcp.2021.110430

19 [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

20 [PINN] 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

21 [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