Learned iterative solvers for the Helmholtz equation

Gabrio Rizzuti*, Ali Siahkoohi, Edmond Chow, and Felix J. Herrmann

Georgia Institute of Technology
Neural nets and iterative schemes in computational maths

Structural likeness between neural networks...

\[ \partial_t u(t) = F(t, u(t)) \quad u(t + \Delta t) \approx u(t) + a(W(t) \ast u(t) + b(t)) \]

[Haber and Ruthotto, 2017]
Neural nets and iterative schemes in computational maths

Structural likeness between **neural networks**…

\[ m \frac{\partial^2 u}{\partial t^2} - \Delta u = f \quad u(t + \Delta t) \approx 2u(t) - u(t - \Delta t) + \Delta t^2 / m (\Delta u(t) + f(t)) \]

…and iterative computational processes such as:

- **time stepping** in finite-differences [Siahkoohi et al., 2018]
Neural nets and iterative schemes in computational maths

Structural likeness between **neural networks**...

\[
\begin{align*}
\min_{m} J(m) \quad m_{i+1} &= m_i - \alpha \nabla_{m_i} J
\end{align*}
\]

...and iterative computational processes such as:

- nonlinear optimization in **inverse problems** [Adler and Öktem, 2017]
Neural nets and iterative schemes in computational maths

Structural likeness between neural networks...

\[ Au = b \quad u_{i+1} = u_i + \sum_{j \leq i} \alpha_j r_j \]

...and iterative computational processes such as:

- **iterative solvers** for linear systems (this talk: Helmholtz equation)
Neural nets and iterative schemes in computational maths

Neural nets: generalizes according to available data, but potentially cheaper

Computational “nets”: general but expensive, pre-“trained” by first principles

This work: (neural net)-augmented physics/maths
Helmholtz equation

Long-standing problem in frequency-domain wave equation based imaging:

- numerical solution of the Helmholtz equation (e.g., discretized by finite-differences):

\[
H[m] = -\omega^2 m - \Delta, \quad H[m]u = f
\]
Classical solution methods:

- **direct methods:** LU factorization (e.g., via nested dissection [George, 1973])

Big-O complexity [Mulder and Plessix, 2002]:

<table>
<thead>
<tr>
<th>Complexity</th>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td># grid points</td>
<td>(n^2)</td>
<td>(n^3)</td>
</tr>
<tr>
<td>factorization</td>
<td>(n_f n^3)</td>
<td>(n_f n^6)</td>
</tr>
<tr>
<td>application</td>
<td>(n_s n_f n^2 \log n)</td>
<td>(n_s n_f n^4 \log n)</td>
</tr>
</tbody>
</table>
Helmholtz equation: classical solution methods

Classical solution methods:

- **iterative methods**: Krylov-subspace schemes for indefinite systems (e.g., GMRES, BiCGStab, ... [Saad, 2003]):

\[
    u_{i+1} = u_i + \sum_{j \leq i} \alpha_j r_j, \quad r_j = f - H[m] u_j
\]

Need **pre-conditioning**!
Helmholtz equation: classical solution methods

Classical solution methods:

- **iterative methods**: Krylov-subspace schemes for **indefinite** systems (e.g., GMRES, BiCGStab, ... [Saad, 2003]):

\[
  u_{i+1} = u_i + \sum_{j \leq i} \alpha_j r_j, \quad r_j = f - H[m] u_j
\]

Need **pre-conditioning**!

E.g., **shifted-Laplacian** preconditioning by multigrid [Erlangga et al., 2006]:

\[
  H_\beta[m] u = f, \quad H_\beta[m] = -\omega^2 (1 - \beta i) m - \Delta
\]

...competitive with time-domain based imaging? # iter grow linearly with frequency [Knibbe et al., 2014]
Helmholtz equation: potential role of machine learning?

Assumptions:

- ultimate goal: solve the inverse problem; we don’t need/want overly “accurate” solutions (even better, solve forward and inverse map jointly?)
- specialized right-hand sides (e.g., point sources)
- prior information about model parameter distribution is often available

Role of machine learning:

- specialize classical methods to a restricted class of problems = accelerate classical methods for the problem at hand
PDE solution by machine learning: general overview

Ever growing body of work, so far focused on learning solutions which generalize over:

- boundary conditions and domain geometry
- right-hand side
- initial conditions; etc...

<table>
<thead>
<tr>
<th>Equation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson equation</td>
<td>[Tang et al., 2017], [Tompson et al., 2017], [Farimani et al. 2017], [Zhang et al., 2018], [Hsieh et al., 2019]</td>
</tr>
<tr>
<td>Laplace equation</td>
<td>[Sharma et al., 2017]</td>
</tr>
<tr>
<td>Schrodinger equation</td>
<td>[Mills et al., 2017]</td>
</tr>
<tr>
<td>Fluid dynamics</td>
<td>[Guo et al., 2016], [Yang et al., 2016], [Chu and Thuerey, 2017], [Kutz, 2017], [Singh et al., 2017]</td>
</tr>
<tr>
<td>Black-Scholes</td>
<td>[Sirignano and Spiliopoulos, 2018]</td>
</tr>
</tbody>
</table>
Goal:

- approximate the Helmholtz solution map with a net-based approximation (for a fixed source and frequency)

\[ F : \mathcal{M} \rightarrow \mathcal{U}, \quad F(m) = (H[m])^{-1} f \iff F_\theta : \mathcal{M} \rightarrow \mathcal{U}, \quad F_\theta(m) \approx (H[m])^{-1} f \]

Candidate loss functions:

- supervised, given solution (this talk):
  \[ L(\theta) = \mathbb{E}_{m \sim p_M} \| F(m) - F_\theta(m) \|_2^2 \]

- unsupervised:
  \[ L(\theta) = \mathbb{E}_{m \sim p_M} \| f - H[m] F_\theta(m) \|_2^2 \]

- Training with stochastic gradient descent algorithms (ADAM, [Kingma and Ba, 2015])
Main idea: **intersperse** Krylov-subspace “nets” and neural nets...

\[ F(\theta_1, \ldots, \theta_N) : \mathcal{M} \to \mathcal{U}, \quad F(\theta_1, \ldots, \theta_N)(\mathbf{m}) = \begin{cases} F_{Kr}^k(\mathbf{m}, N_{\theta_N} \circ F(\theta_1, \ldots, \theta_{N-1})(\mathbf{m})), & N \geq 1 \\ F_{Kr}^k(\mathbf{m}, [\mathbf{u}_0]), & N = 0 \end{cases} \]

\[ F_{Kr}^k : \mathcal{M} \times \mathcal{U} \to \mathcal{U}, \quad F_{Kr}^k(\mathbf{m}, \mathbf{u}) = \mathbf{u} + \sum_{j=0}^{k-1} \alpha_j H[\mathbf{m}]^j \mathbf{r}, \text{ for some } \alpha_j \quad (\in \mathbf{u} + \text{Kr}(H[\mathbf{m}], \mathbf{r})) \]

\[ N_{\theta_i} : \mathcal{U} \to \mathcal{U} \]
2-D net correction architecture ("Unet") ~ multigrid (e.g. [Ke et al., 2017], [He and Xu, 2019]):

Level 1: input size $n^2$, 1 ch

Level 2: input $(n/2)^2$, 2 ch

Level 3: input $(n/4)^2$, 4 ch

skip connection

skip connection

resnet [He et al., 2015]

Complexity $\sim O(n^2 \log n)$
2-D net correction architecture, \textbf{two-grid sketch}:

Pre-smoothing (fine grid):
\[
x^h \leftarrow x^h + a(W_k^h \ast x^h + b_k^h), \quad \text{for } k = 1, \ldots, N
\]

Restriction (many channels!):
\[
x_{ch_i}^{2h} \leftarrow R_i^{2h}(W_{ch_i}^h \ast x^h + b_{ch_i}^h), \quad \text{for } i = 1, \ldots, N_{ch}
\]

Smoothing (coarse grid):
\[
x_{ch_i}^{2h} \leftarrow x_{ch_i}^{2h} + a(\sum_j W_{k, ch_i, ch_j}^{2h} \ast x_{ch_j}^{2h} + b_{k, ch_i}^{2h}), \quad \text{for } i, k, \ldots
\]

Prolongation:
\[
x^h \leftarrow x^h + P_{2h}^h \sum_i (W_{ch_i}^{2h} \ast x_{ch_i}^{2h} + b_{ch_i}^{2h})
\]

Post-smoothing (fine grid):
\[
x^h \leftarrow x^h + a(W_l^h \ast x^h + b_l^h), \quad \text{for } l = 1, \ldots, N
\]
Example 1: layered model distribution (fixed source and frequency)

\[ z \sim z^0 + U(-a, a) \quad \theta \sim U(-b, b) \]
Example 1: layered model distribution
(train size: 1024, test size: 16)
Example 1: solution distribution at 60 Hz (train size: 1024, test size: 16)
Example 1: approximated wavefield at 60 Hz (after 5 Krylov iterations)
Example 1: approximated wavefield at 60 Hz (after 10 Krylov iterations)
Example 1: approximated wavefield at 60 Hz (after 15 Krylov iterations)
Example 1: approximated wavefield at 60 Hz (after 20 Krylov iterations)
Example 1: approximated wavefield at 60 Hz (after 25 Krylov iterations)
Example 1: solution distribution at 60 Hz (train size: 1024, test size: 16)
Example 1: approximated wavefield at 60 Hz (after 5 Krylov iterations + net)
Example 1: approximated wavefield at 60 Hz (after 10 Krylov iterations + net)
Example 1: approximated wavefield at 60 Hz (after 15 Krylov iterations + net)
Example 1: approximated wavefield at 60 Hz (after 20 Krylov iterations + net)

Solution after 20 Krylov iterations + net correction
Example 1: approximated wavefield at 60 Hz (after 25 Krylov iterations \(+ net\))

Solution after 25 Krylov iterations + net correction

Wavefield # 1

Wavefield # 4

Wavefield # 7

Wavefield # 2

Wavefield # 5

Wavefield # 8

Wavefield # 3

Wavefield # 6

Wavefield # 9
Example 1: solution trace comparison
Example 1: training/test errors

\[
L = \sqrt{\sum_i \|u_i^{sol} - \tilde{u}_i\|^2 / \sum_i \|u_i^{sol}\|^2}
\]

<table>
<thead>
<tr>
<th></th>
<th>Train error</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krylov iterations</td>
<td>23%</td>
<td>24%</td>
</tr>
<tr>
<td>Krylov net</td>
<td>3.9%</td>
<td>5.2%</td>
</tr>
</tbody>
</table>
Example 2: Marmousi-like distribution (fixed source and frequency)

\[ z_c \sim U(z_0, z_{\text{end}}) \quad w \sim w_0 + U(-a, a) \]
Example 2: Marmousi-like distribution (train size: 1024, test size: 16)
Example 2: solution distribution at 60 Hz
(train size: 1024, test size: 16)
Example 2: solution wavefield at 60 Hz (after 5 Krylov iterations)
Example 2: solution wavefield at 60 Hz (after 10 Krylov iterations)
Example 2: solution wavefield at 60 Hz (after 15 Krylov iterations)
Example 2: solution wavefield at 60 Hz (after 20 Krylov iterations)
Example 2: solution wavefield at 60 Hz (after 25 Krylov iterations)
Example 2: solution distribution at 60 Hz (train size: 1024, test size: 16)
Example 2: solution wavefield at 60 Hz (after 5 Krylov iterations + net)
Example 2: solution wavefield at 60 Hz (after 10 Krylov iterations + net)
Example 2: solution wavefield at 60 Hz (after 15 Krylov iterations + net)

Solution after 15 Krylov iterations + net correction
Example 2: solution wavefield at 60 Hz (after 20 Krylov iterations + net)
Example 2: solution wavefield at 60 Hz (after 25 Krylov iterations + net)
Example 2: solution trace comparison

Solution comparison at $z = z_{src}$ (model # 1)

Solution comparison at $x = x_{src}$ (model # 1)
Example 2: testing generalization to different model distributions

\[ z_c \sim U(z_0, z_{\text{end}}) \quad w \sim w_0 + U(-a, a) \]
Example 2: Marmousi-like distribution (generalization test size: 16)
Example 2: Train/test errors

\[
L = \sqrt{\sum_i \|u_i^{\text{sol}} - \ddot{u}_i\|^2 / \sum_i \|u_i^{\text{sol}}\|^2}
\]

<table>
<thead>
<tr>
<th></th>
<th>Train error</th>
<th>Test error</th>
<th>“Generalization” error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krylov iterations</td>
<td>37.6%</td>
<td>40.5%</td>
<td>32.3%</td>
</tr>
<tr>
<td>Krylov net</td>
<td>12.1%</td>
<td>12.9%</td>
<td>17.2%</td>
</tr>
</tbody>
</table>
Possible improvements:

- **GANs**

\[
D_\varphi : \mathcal{U} \rightarrow [0, 1] \quad \text{discriminator}
\]

\[
L(\theta, \varphi) = \mathbb{E}_{u \sim p_U} (1 - D_\varphi(u))^2 + \mathbb{E}_{m \sim p_M} (D_\varphi \circ F_\theta(m))^2 + \lambda \mathbb{E}_{m, u \sim p_{M, U}} \|u - F_\theta(m)\|^2
\]
Possible improvements:

- GANs
- transfer learning: fine-tune net on a new model distribution
Conclusions/Future plans

Possible improvements:

- GANs
- transfer learning: fine-tune net on a new model distribution
- neural net architecture: inject linear operator residuals at each level and learn restriction/prolongation to beat indefiniteness

\[ x^h \leftarrow x^h + N^h_\theta(r^h) \]
Possible improvements:

- GANs
- transfer learning: fine-tune net on a new model distribution
- neural net architecture: inject linear operator residuals at each level and learn restriction/prolongation to beat indefiniteness
- multiscale loss function for unsupervised case

\[
L = \sum_{j} \left\| R_{h}^{j} (f - H[m] F_{\theta}(m)) \right\|^2
\]
Conclusions/Future plans

Possible improvements:

- GANs
- transfer learning: fine-tune net on a new model distribution
- neural net architecture: inject linear operator residuals at each level and learn restriction/prolongation to beat indefiniteness
- multiscale loss function for unsupervised case

Alternative applications/extensions:

- implicit time-stepping
- source-to-source / low-to-high frequency / acoustic-to-elastic transfer
- combination with learned reconstruction operators
<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guo, X., W. Li, and F. Iorio, Convolutional Neural Networks for Steady Flow Approximation, KDD (2016)</td>
</tr>
<tr>
<td>Ke, T.-W., M. Maire, and S. X. Xu, Multigrid Neural Architectures, CVPR (2017)</td>
</tr>
<tr>
<td>Knibbe, H., W.A. Mulder, C.W. Oosterlee, C. Vuik, Closing the performance gap between an iterative frequency-domain solver and an explicit time-domain scheme for 3D migration on parallel architectures, Geophysics (2014)</td>
</tr>
</tbody>
</table>


