جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

The trials and errors of physically informing neurons

Physics informed neural networks (PINNs)

incorporate physical laws, represented by partial differential equations (PDEs), into the loss function of a neural network. This allows for solving forward and inverse problems in physics.

Why Use PINNs?

- Enforce physics constraints without large labeled datasets -- unsupervised
- Solve PDEs efficiently?
- Useful in fluid dynamics, material science, electromagnetics, seismic waves etc.
- Provide better generalization than purely data-driven models

Partial differential equation (PDE)

is a type of mathematical equation that involves partial derivatives of an unknown function with respect to multiple variables. PDEs are used to describe various physical phenomena, such as heat conduction, wave propagation, fluid dynamics, and quantum mechanics.

Mathematical Formulation

- Consider a physical problem, and the corresponding PDE
- *PDE(u(**x**,t)) = 0*, where *u(**x**,t)* is the solution
- PINNs approximate *u(**x**,t)* using an NN, $u_\theta(\mathbf{x},t)$. The residual of the PDE is minimized in the loss function.

Stress and Motion Equations

- Equation of Motion (Newton's Second Law):

  $\rho \, \partial^2 u/\partial t^2 = \nabla \cdot \sigma$

  - where ρ is density, u is displacement, and σ is the stress tensor.
- Constitutive Equation (Hooke's Law):

  $\sigma = C : \varepsilon$

  - where C is the stiffness tensor, and ε is the strain tensor.
- Strain-Displacement Relation:

  $\varepsilon = (\nabla u + (\nabla u)^T)/2$

The Elastic Wave Equation

- Substituting Hooke's law and strain relations:

$$\rho \, \partial^2 u / \partial t^2 = \nabla \cdot (C : \nabla u)$$

- Special cases:
- Isotropic medium: Leads to P-wave and S-wave equations.
- Applications: Seismology, nondestructive testing, material science.

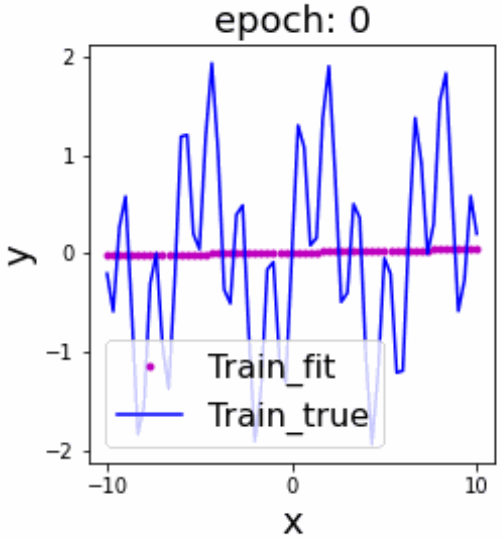
Universal approximation theorem

The approximation capabilities of the feedforward architecture on the space of continuous functions between two Euclidean spaces

$$f_\epsilon : \mathbb{R}^d \to \mathbb{R}^D \text{ with representation}$$

$$f_\epsilon = W_2 \circ \sigma \bullet W_1,$$

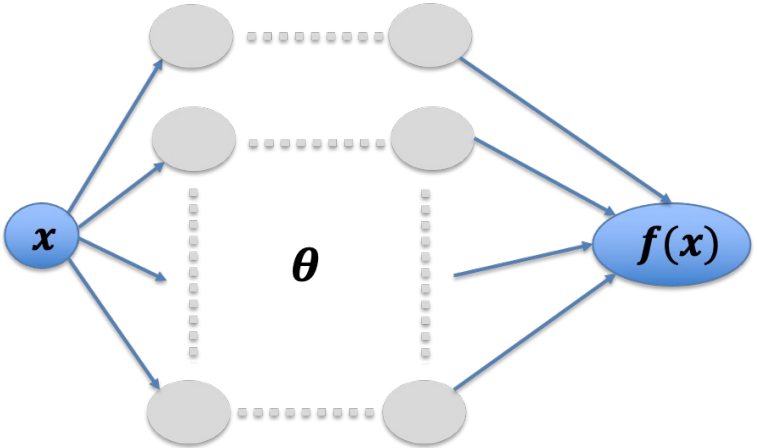
$$\sup_{x \in K} \|f(x) - f_\epsilon(x)\| < \varepsilon$$



Pinkus (1999)

KAUST

The framework (Raissi et al., 2019)



$$\boldsymbol{Loss} = |\boldsymbol{PDE}(\boldsymbol{f}(\boldsymbol{x};\ \boldsymbol{\theta}), a)|$$

A fully connected MLP:

$u_\theta(\mathbf{x}, t) = \sigma_L(W_L\, \sigma_{L-1}(\ldots\sigma_1(W_1\, \mathbf{x} + b_1)\ldots) + b_L)$

- Nonlinear activation functions improve expressiveness.
- Last layer we often use linear activation functions

The loss function

$$MSE = MSE_u + MSE_f$$

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

Raissi et.al. (2019)

Automatic Differentiation (AD)

- AD efficiently (?) computes derivatives using the chain rule:

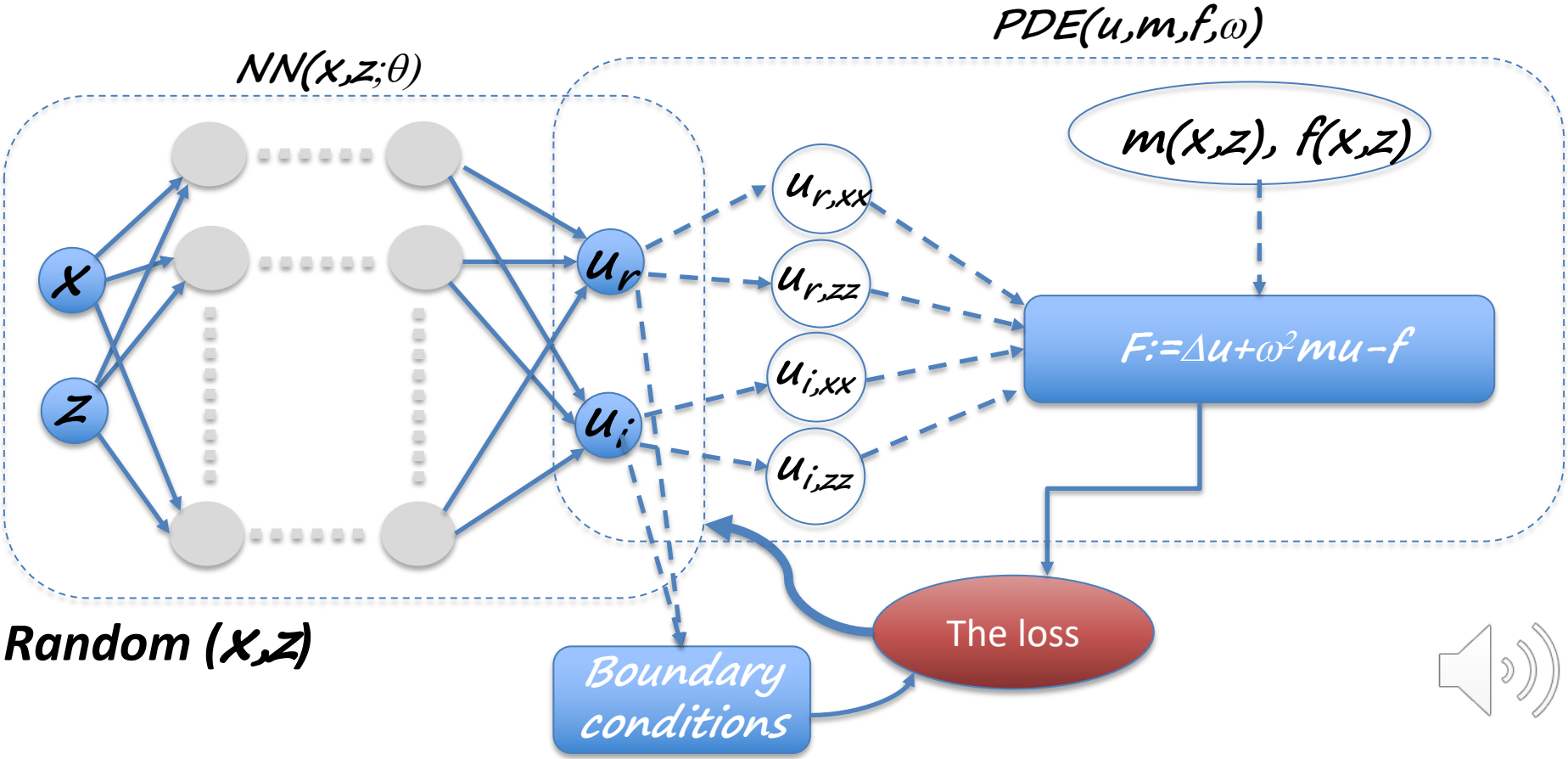
$$df/dx = \sum_i \partial f/\partial h_i \, \partial h_i/\partial x$$

- Enables enforcement of PDE constraints.

Training Process

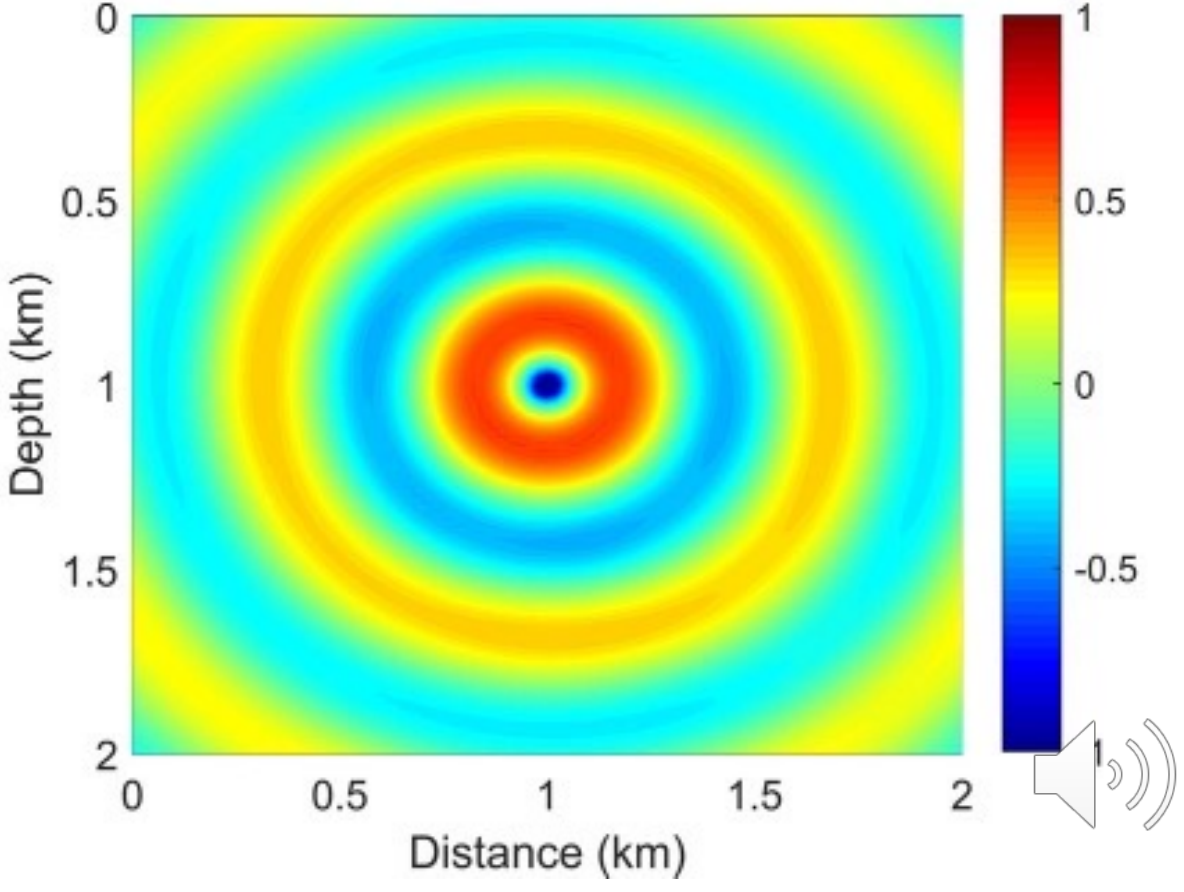
- Define neural network $u_\theta(\mathbf{x},t)$, usually an MLP
- Pick points from the solution domain to train the network
- Measure the loss through Automatic Differentiation (AD)
- Minimize loss using gradient-based optimization
- Evaluate accuracy with analytical/numerical solutions

The network (Helmholtz equation)

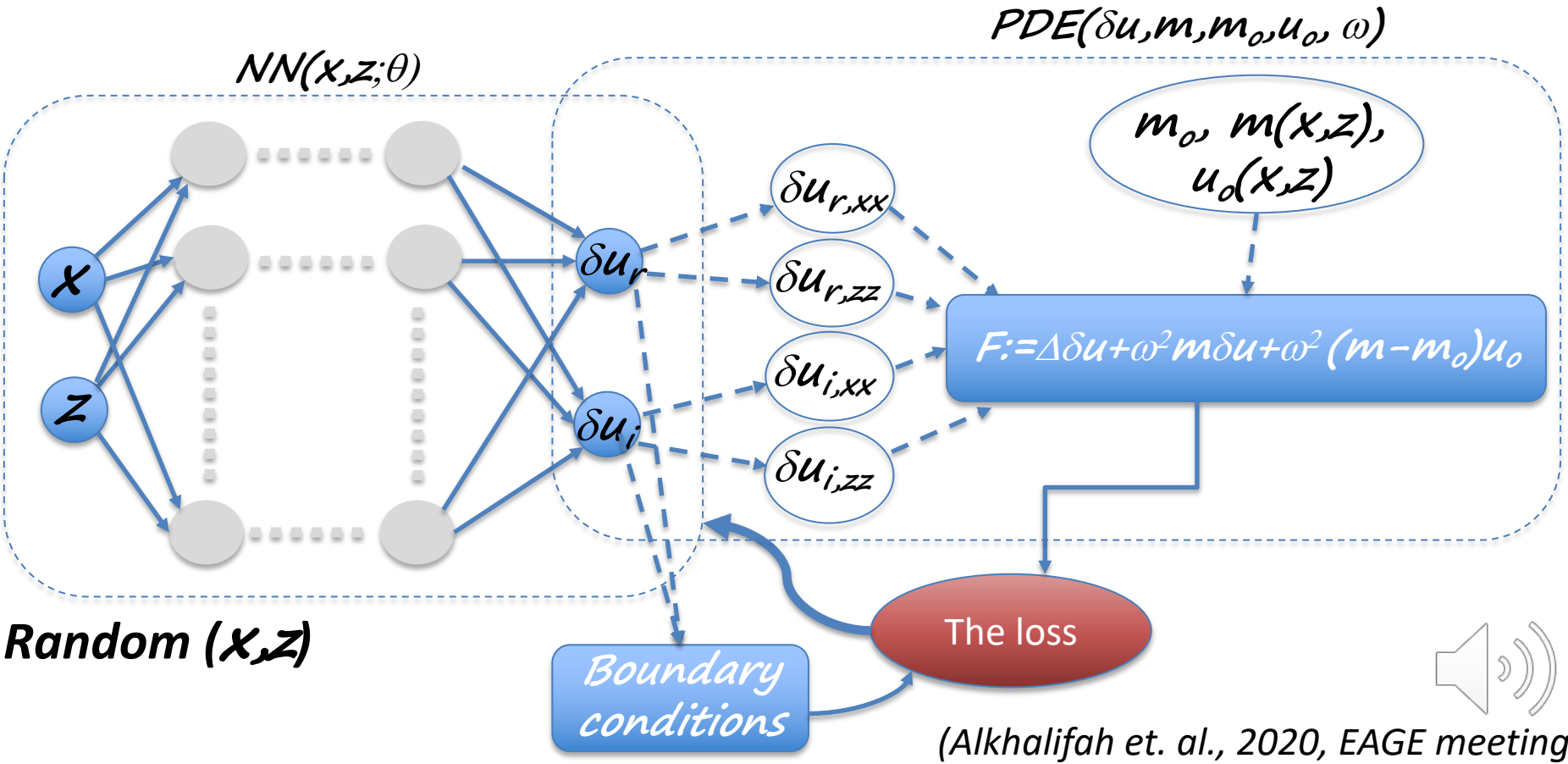


The source singularity

Requires more samples near the source, where the gradient is not defined



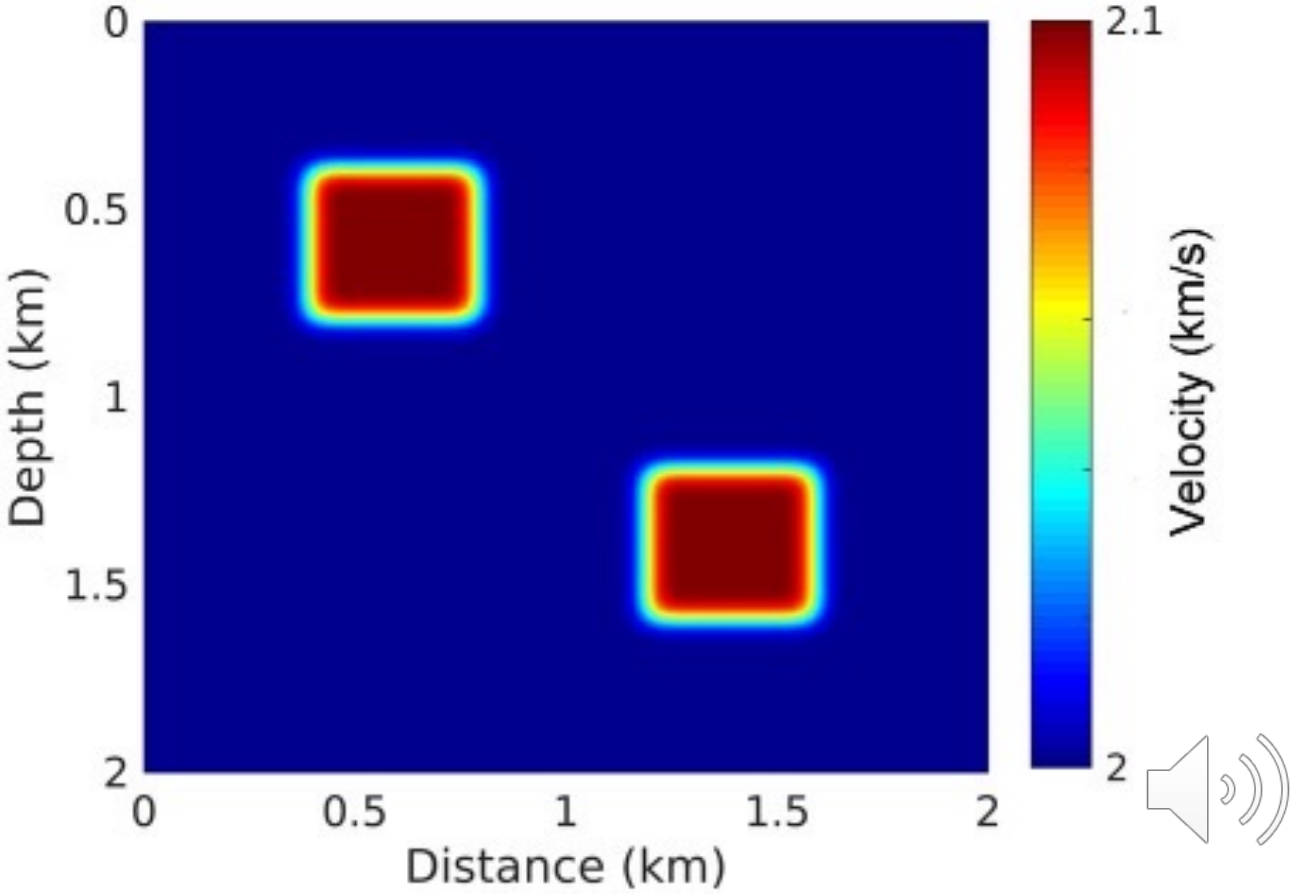
The scattered wavefield



(Alkhalifah et. al., 2020, EAGE meeting)

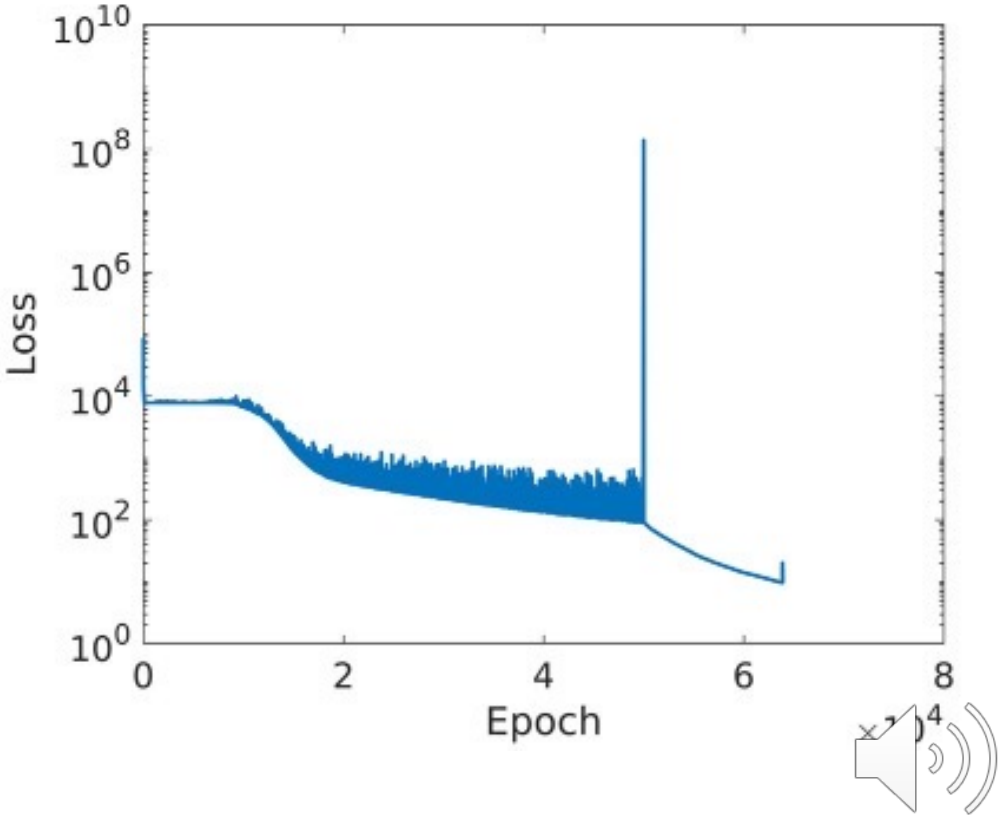
Two box model, homogenous background

The background velocity is 2 km/s

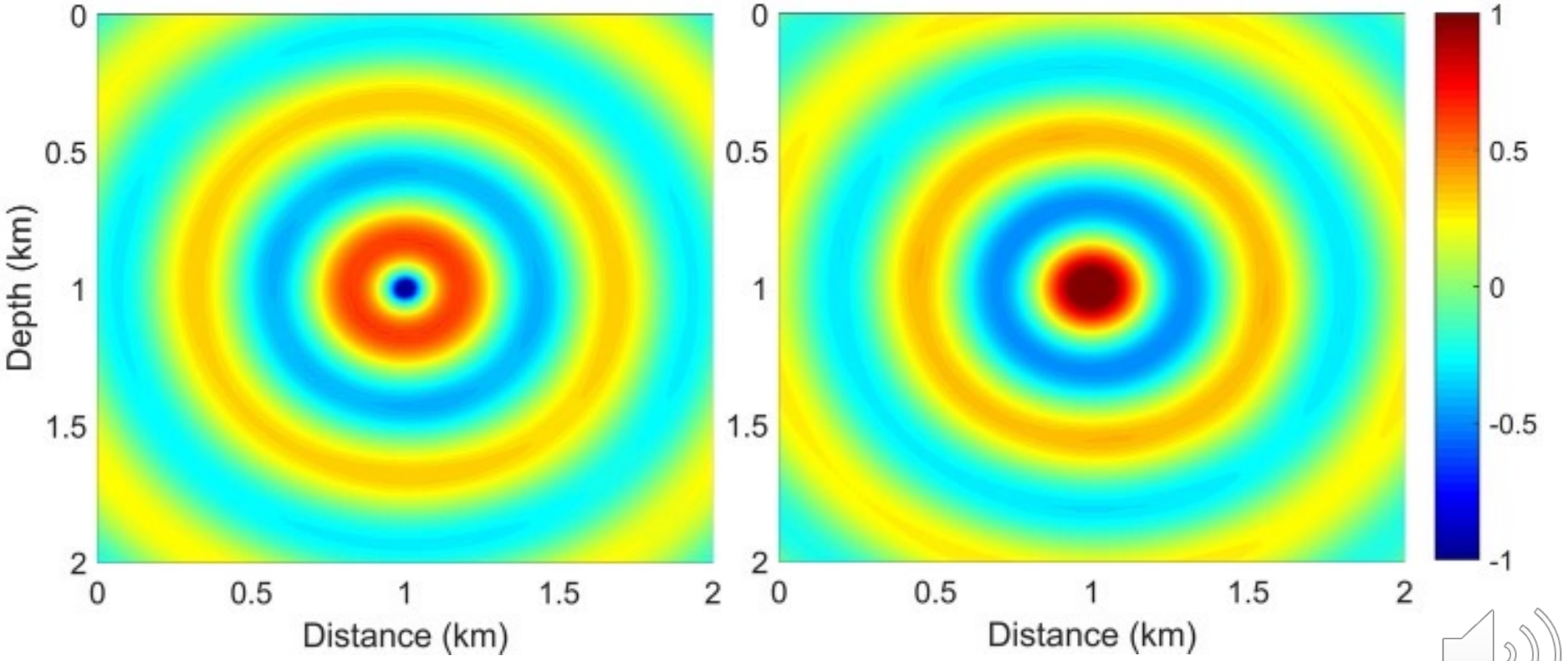


The training (4Hz)

- *8-hidden-layers, 20 neurons*
- *5000 random samples*
- *Full batch training*
- *50000 Adam iterations*
- *20000 l-BFGS iterations*
- *Inverse tangent activation function here and throughout*



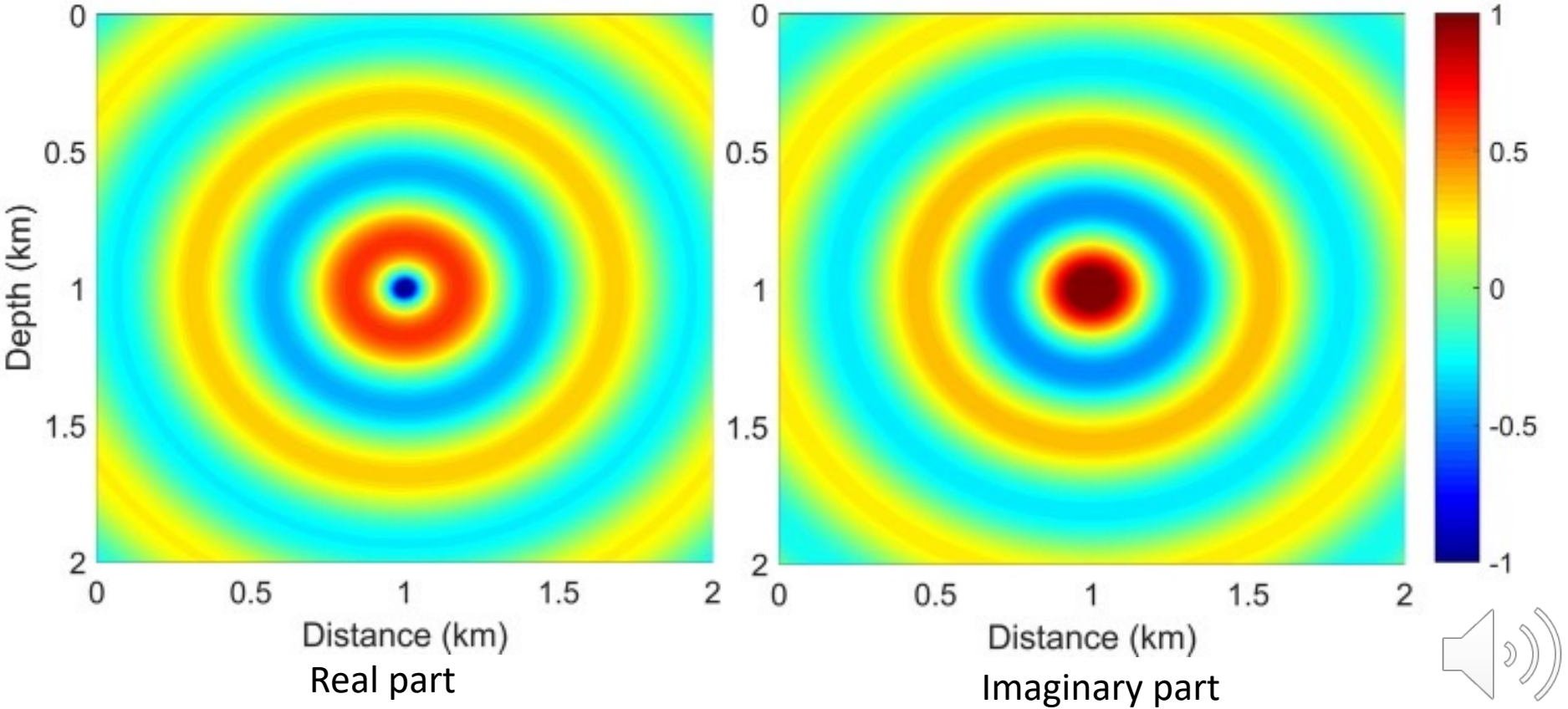
The numerical 4 Hz wavefield



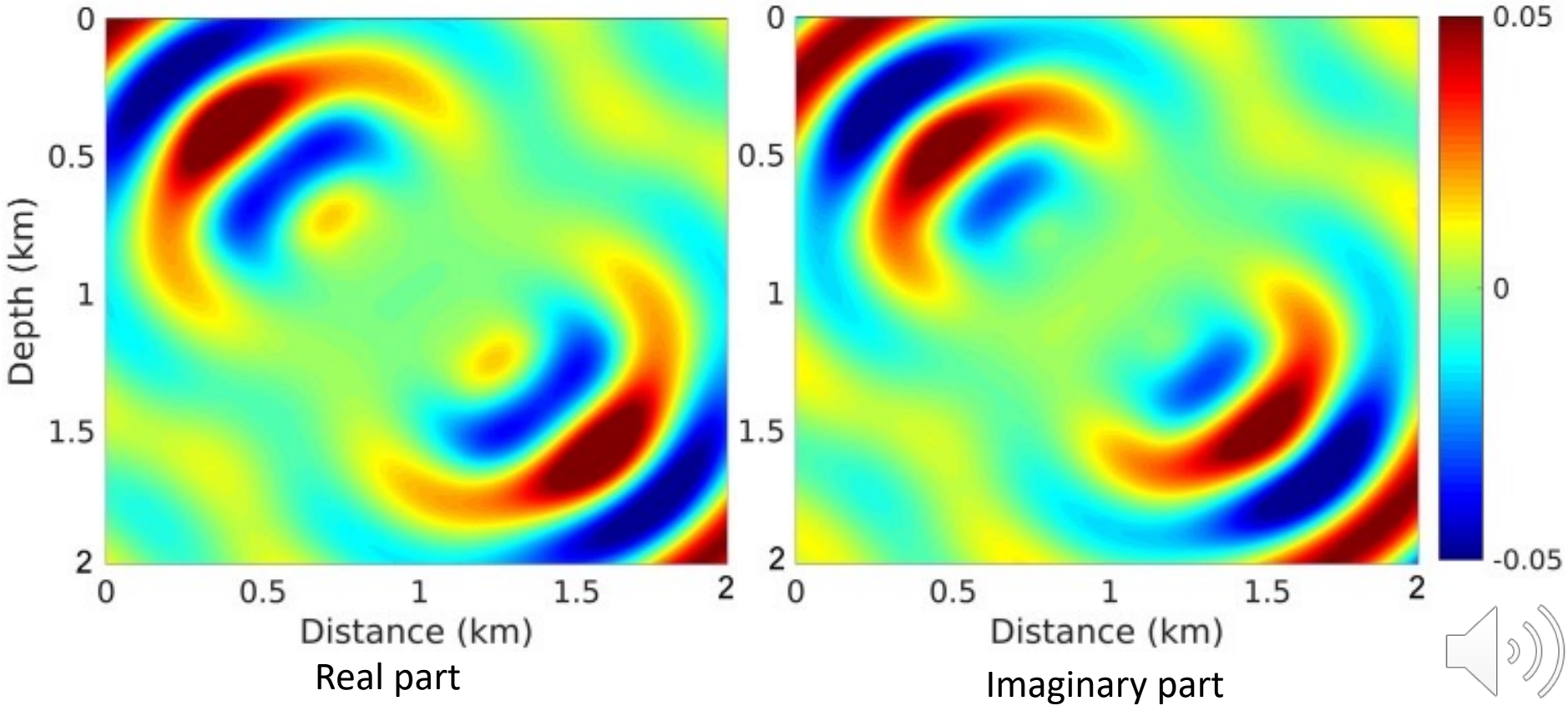
Real part

Imaginary part

4 Hz wavefield, homogeneous background 2 km/s



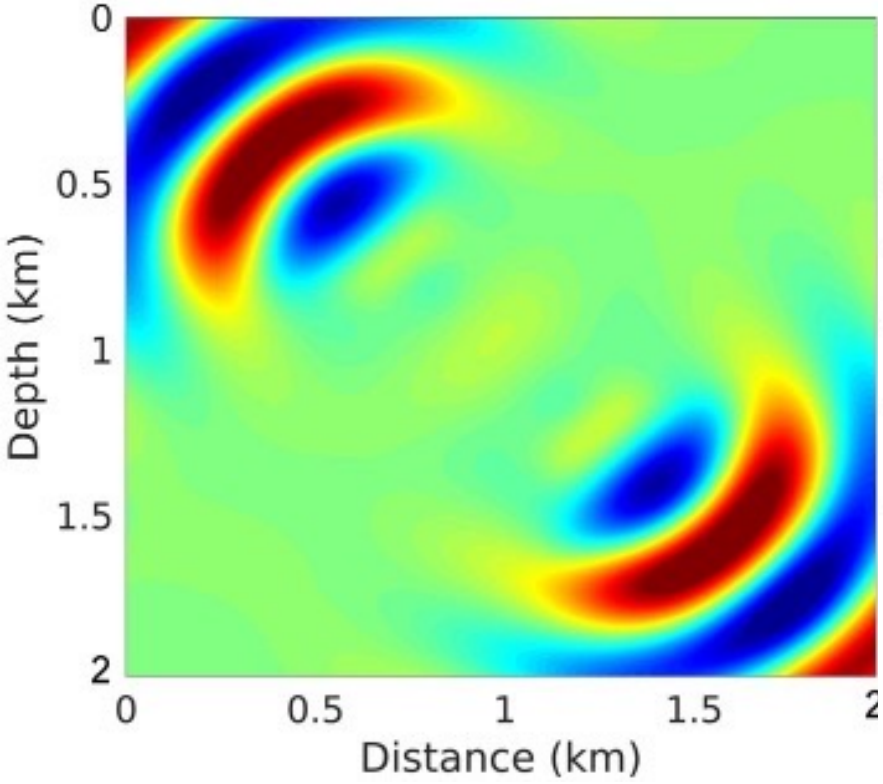
The numerical 4 Hz scattered wavefield



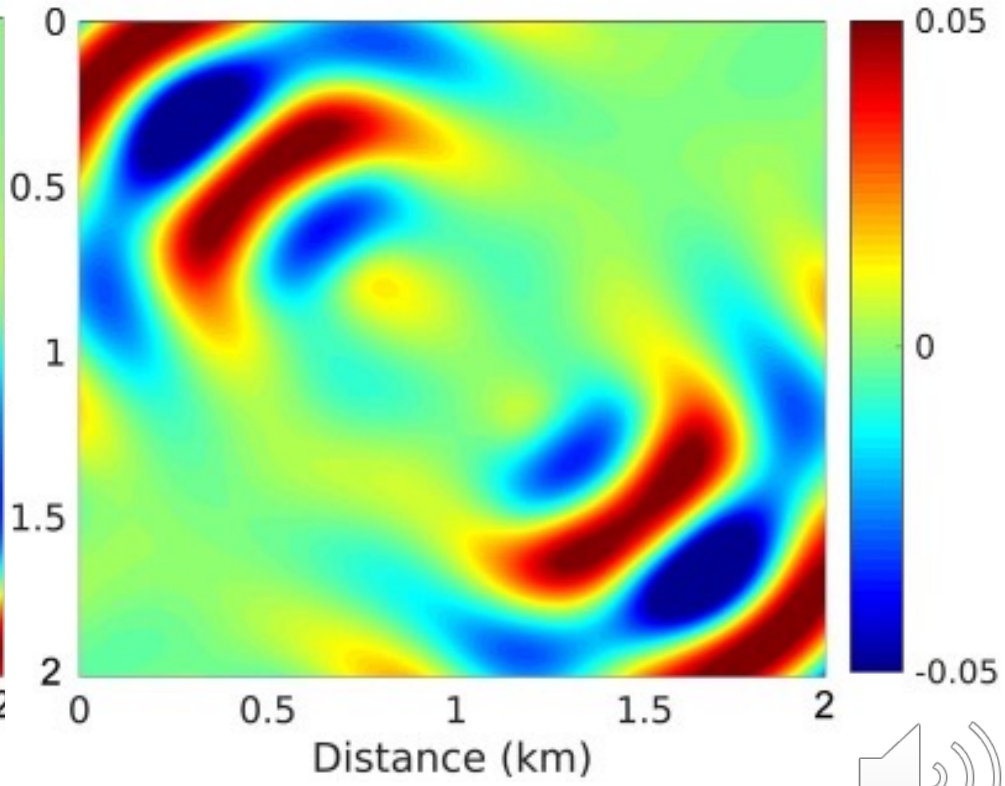
Real part

Imaginary part

The NN 4 Hz scattered wavefield

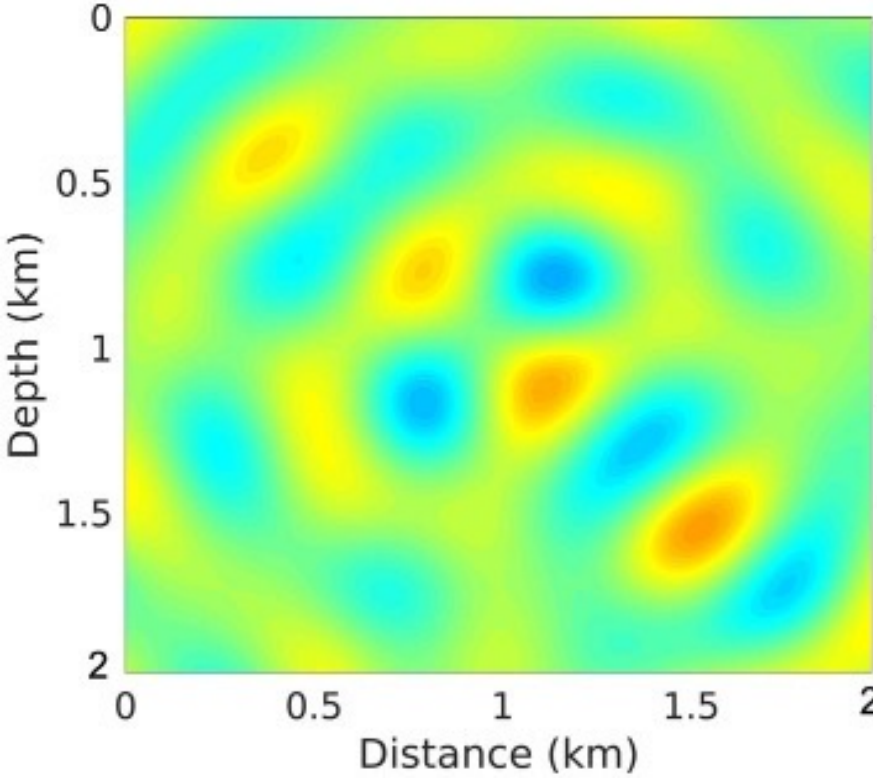


Real part

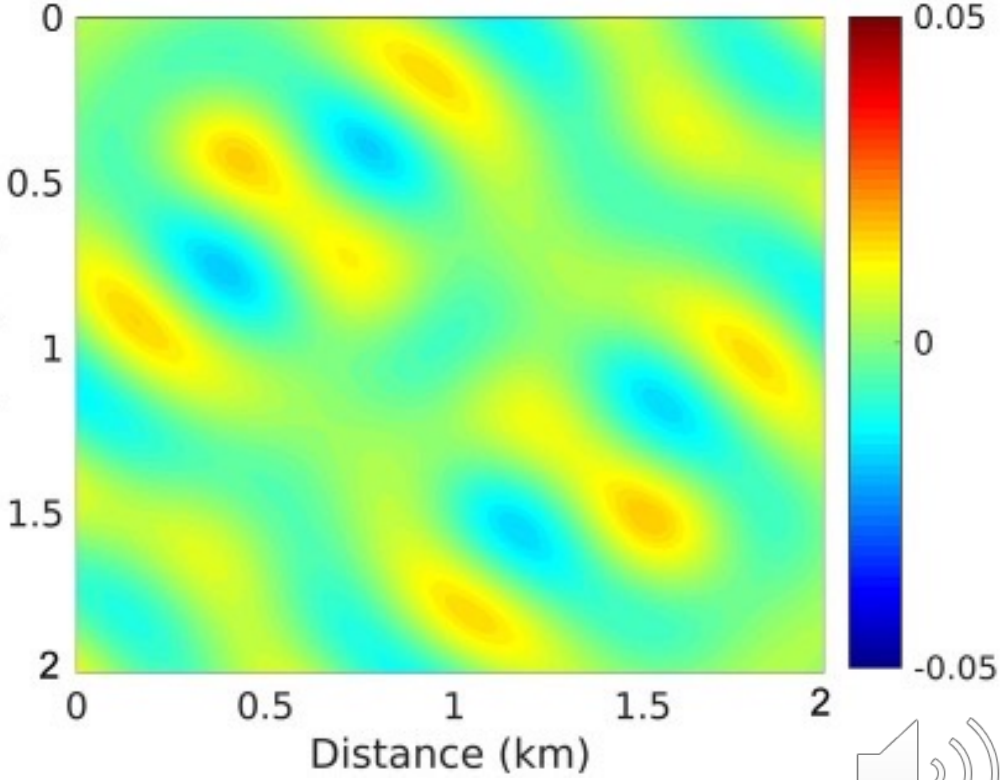


Imaginary part

The difference, plotted at the same scale



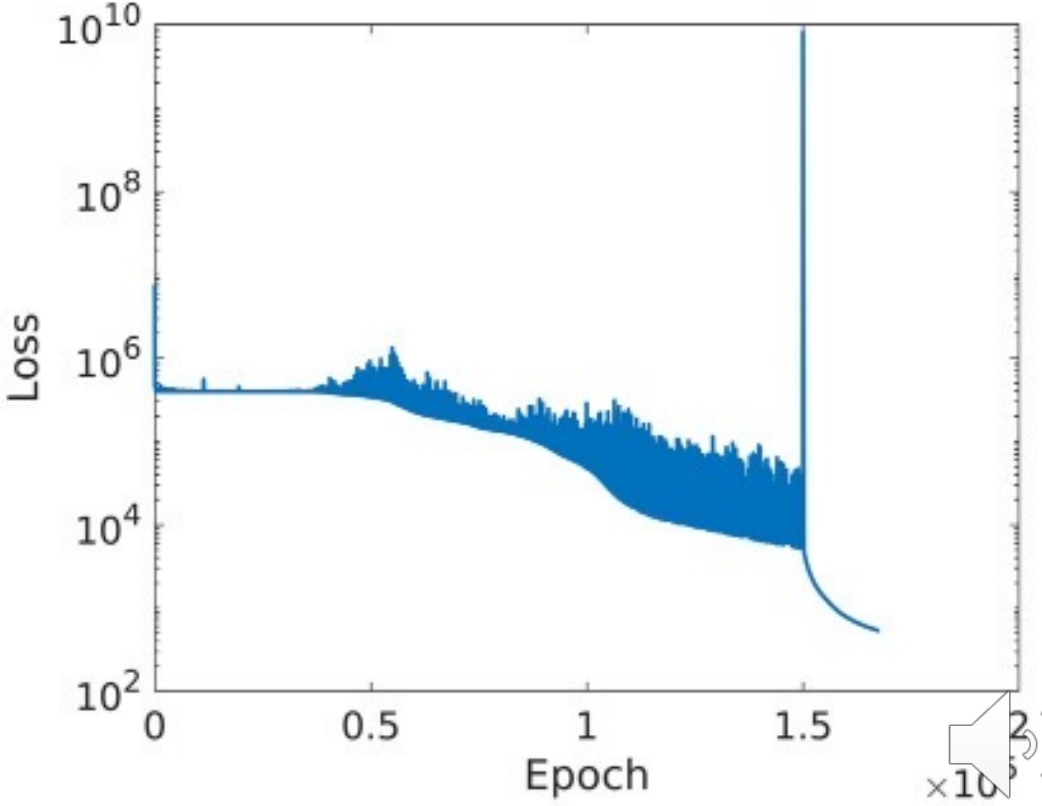
Real part



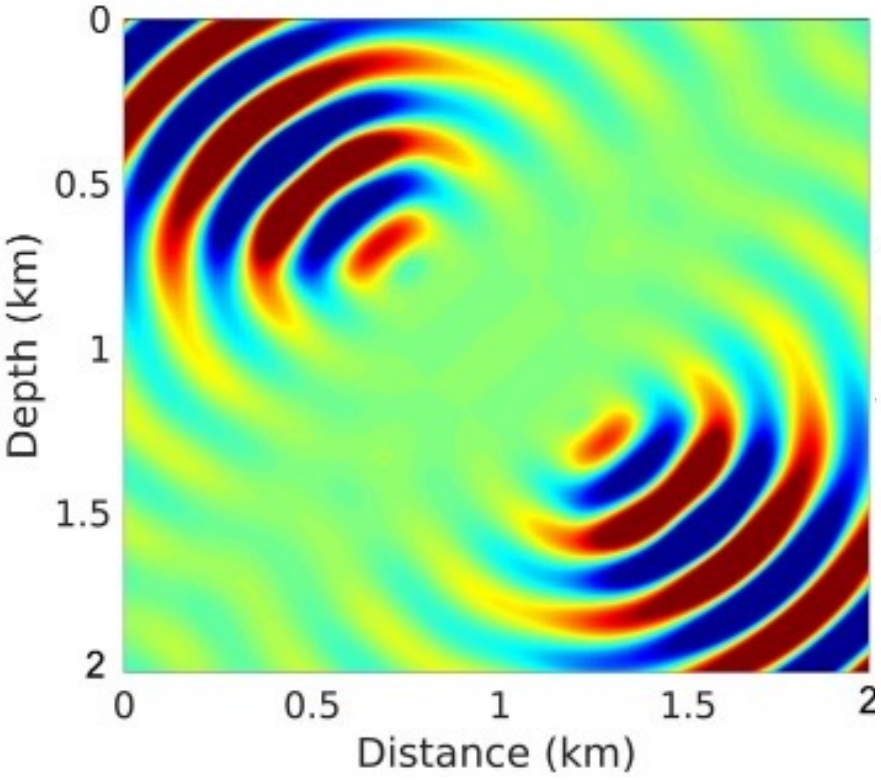
Imaginary part

The training (8Hz)

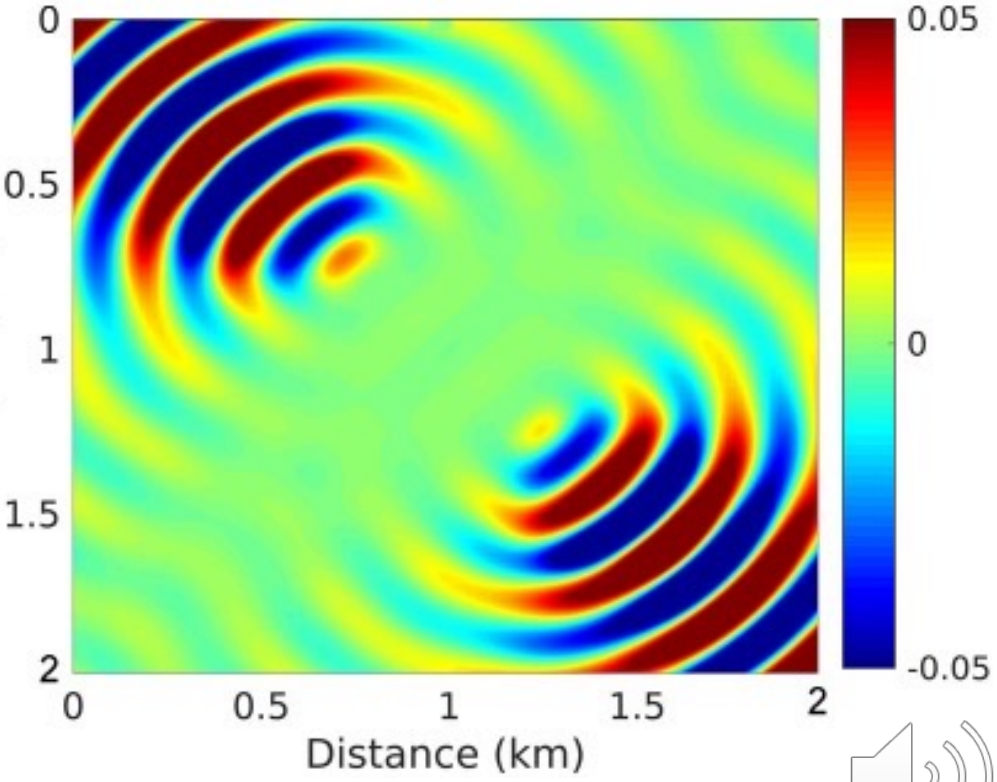
- *8-layers, 40 neurons*
- *10000 random samples*
- *Full batch training*
- *150000 Adam iterations*
- *20000 l-BFGS iterations*
- *Cost increase over the 4Hz is about 100%*



The numerical 8 Hz scattered wavefield

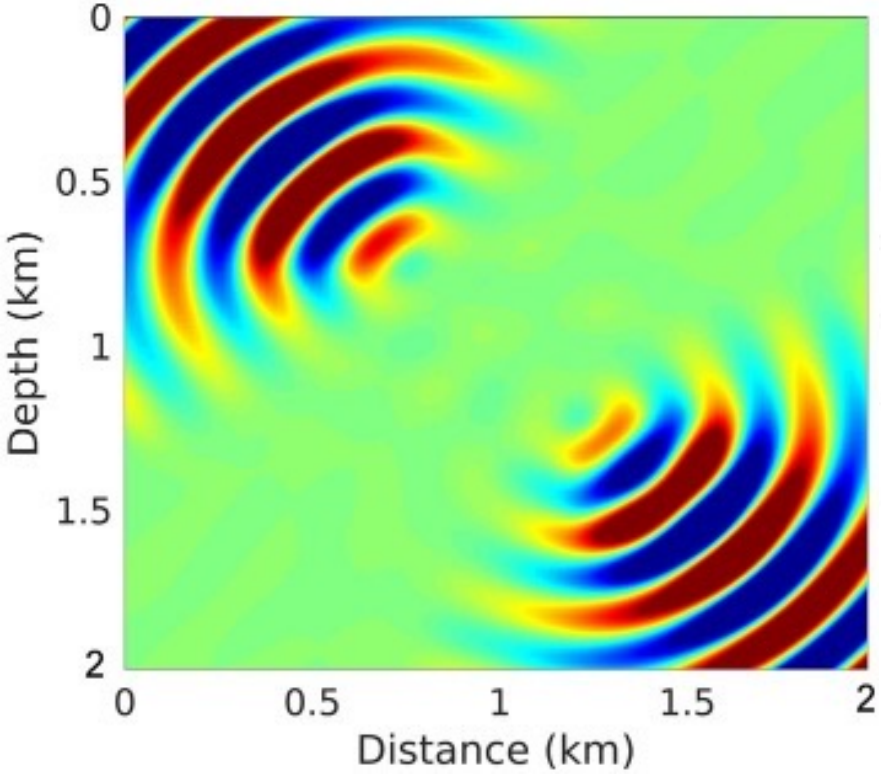


Real part

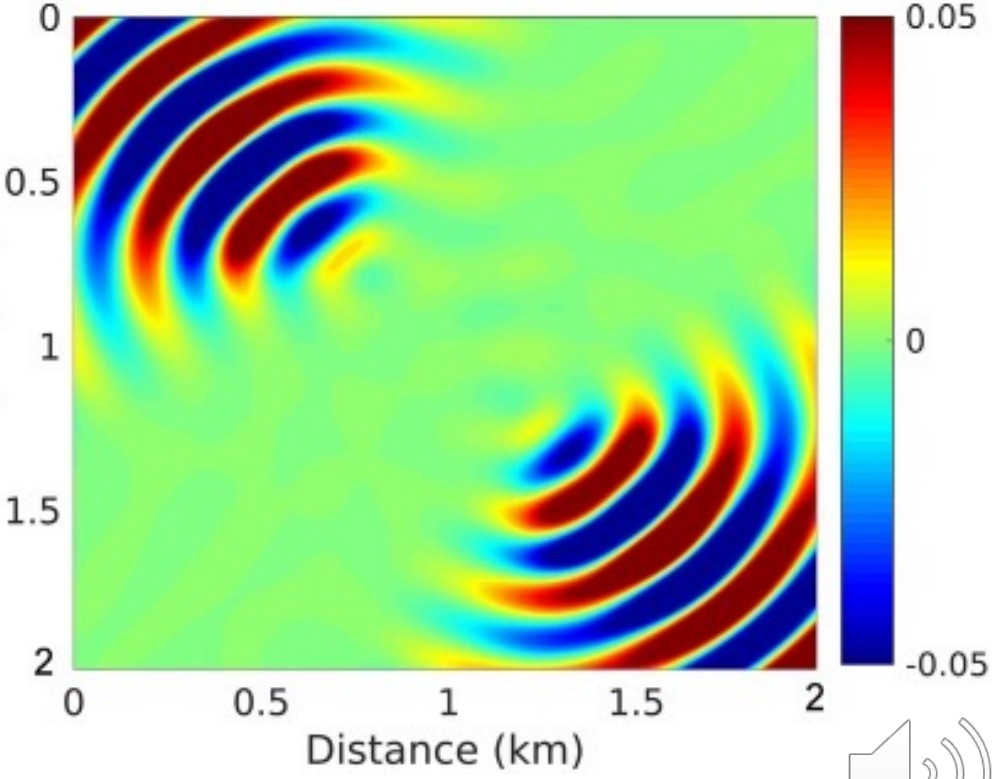


Imaginary part

The NN 8 Hz scattered wavefield

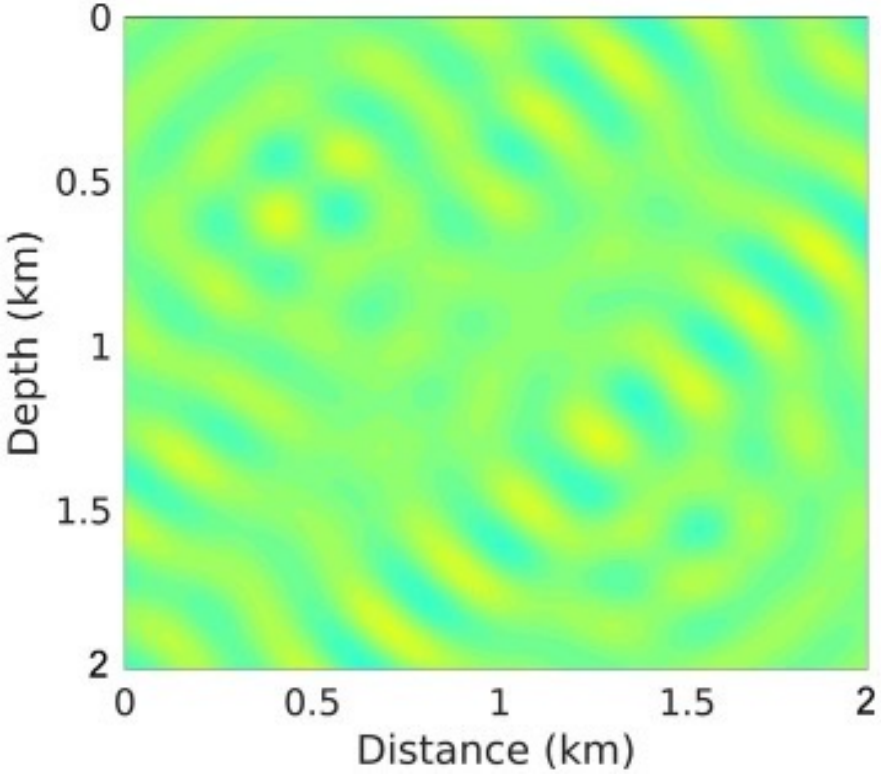


Real part

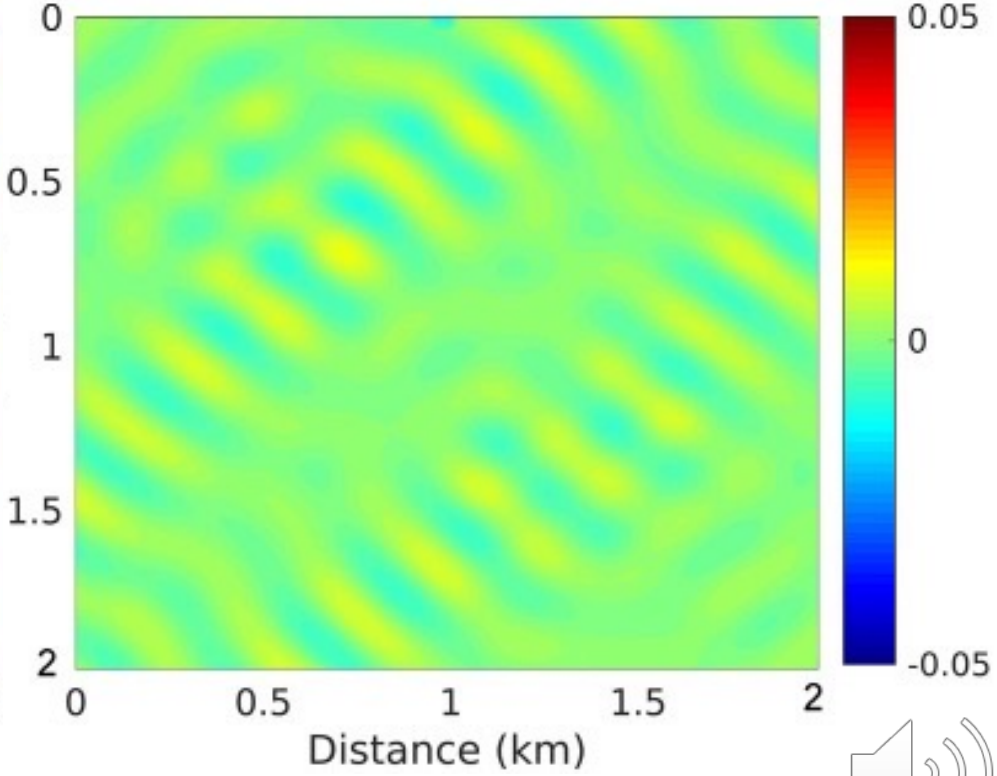


Imaginary part

The difference, plotted at the same scale



Real part



Imaginary part

Green's functions

- The inverse of the impendence matrix

$$\left(\nabla^2 + \frac{\omega^2}{v^2}\right) G(\mathbf{x};\mathbf{x_s}) = \delta(\mathbf{x}-\mathbf{x_s})$$

$$\delta G = G - G_0 \qquad \delta m = \frac{1}{v^2} - \frac{1}{v_0^2}$$

$$\left(\nabla^2 + \frac{\omega^2}{v^2}\right) \delta G(\mathbf{x};\mathbf{x_s}) = -\omega^2 \delta m\, G_0(\mathbf{x};\mathbf{x_s})$$

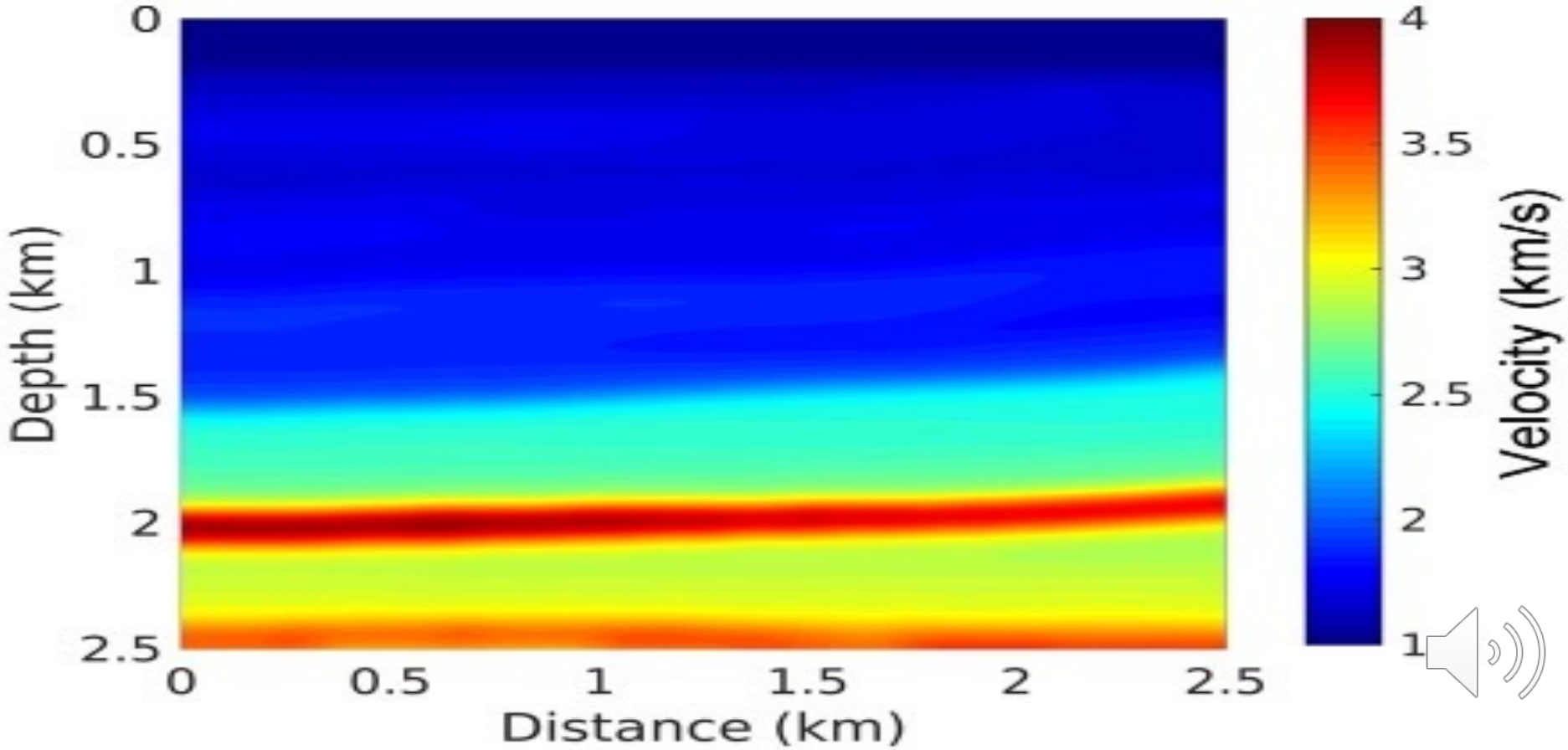
The Loss function

$$f = \frac{1}{N}\sum_{i=1}^{N}\left|\omega^2 m^{(i)}\delta G^{(i)} + \nabla^2\delta G^{(i)} + \omega^2\delta m^{(i)}G_0^{(i)}\right|_2^2$$

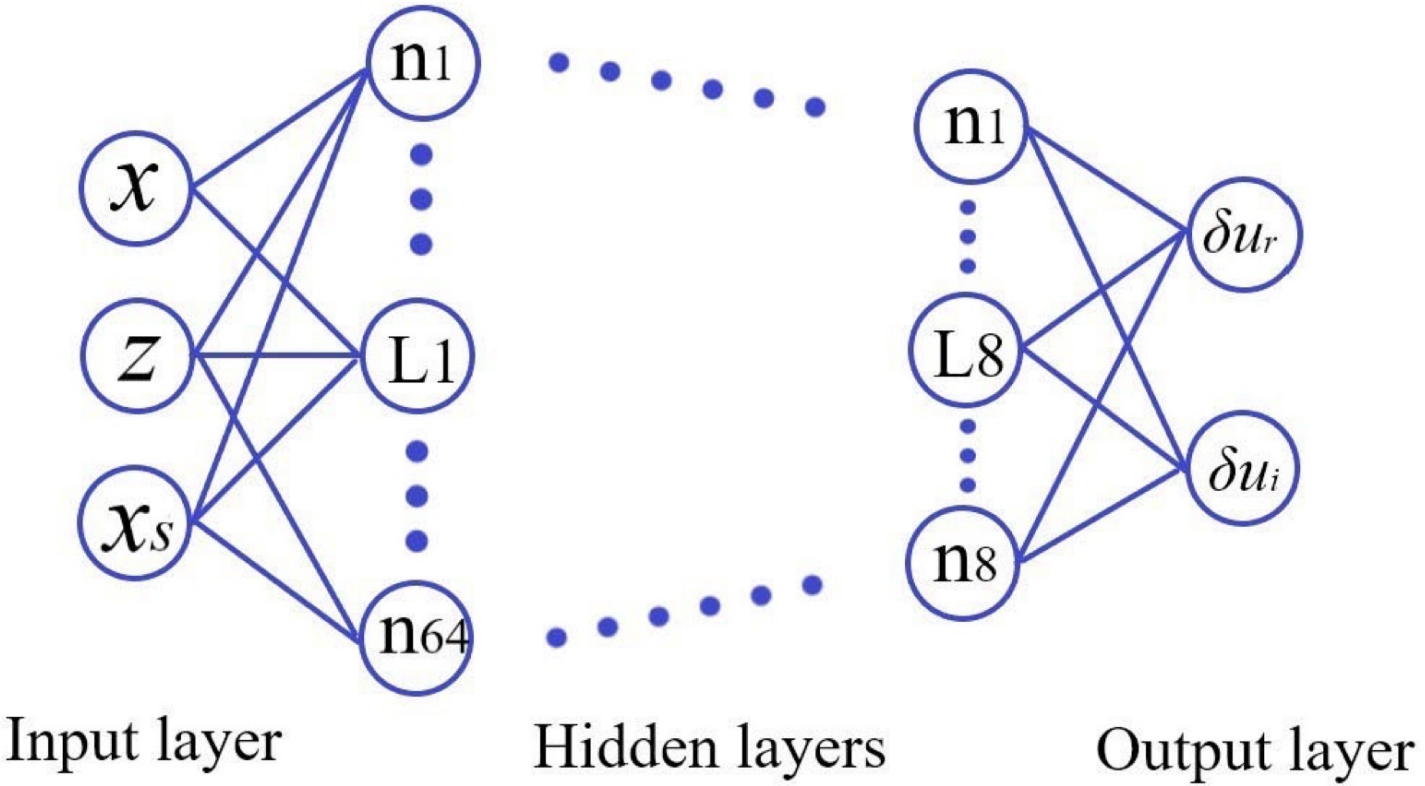
Inputs to the network: x, x_s

In 2D, considering surface sources: x, z, x_s

The velocity model



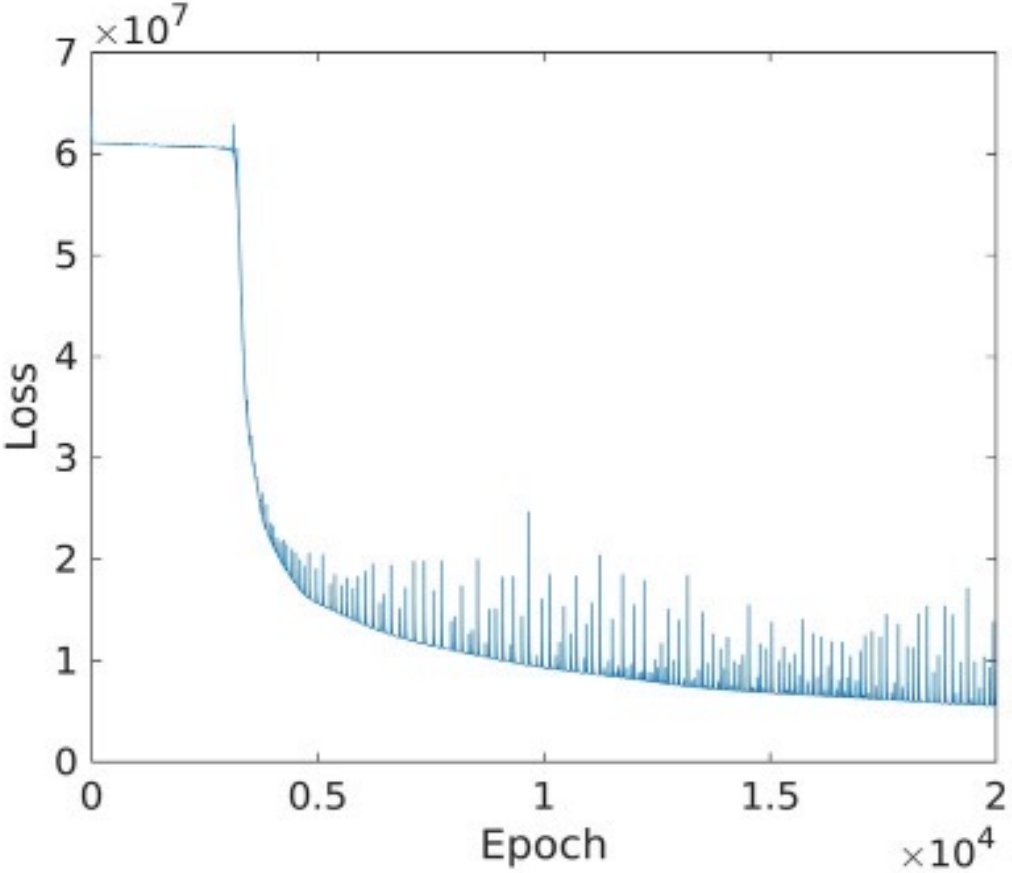
The network



$\delta G(x,z;x_s)$

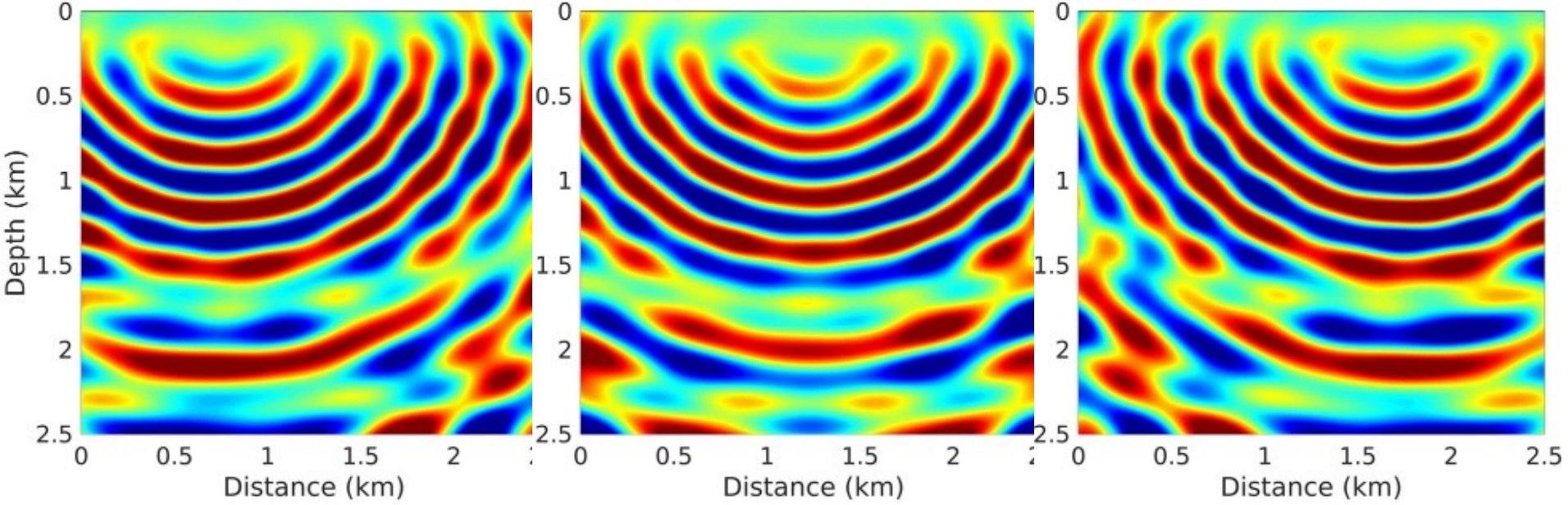
The training

- *5Hz scattered wavefield*
- *8-layers*
- *{64,64,32,32,16,16,8,8}*
- *10000 random samples in (x,z,x_s)*
- *Full batch training*
- *20000 Adam iterations*



10000 random training samples (x,z,x_s)

The numerical scattered wavefield (real part)

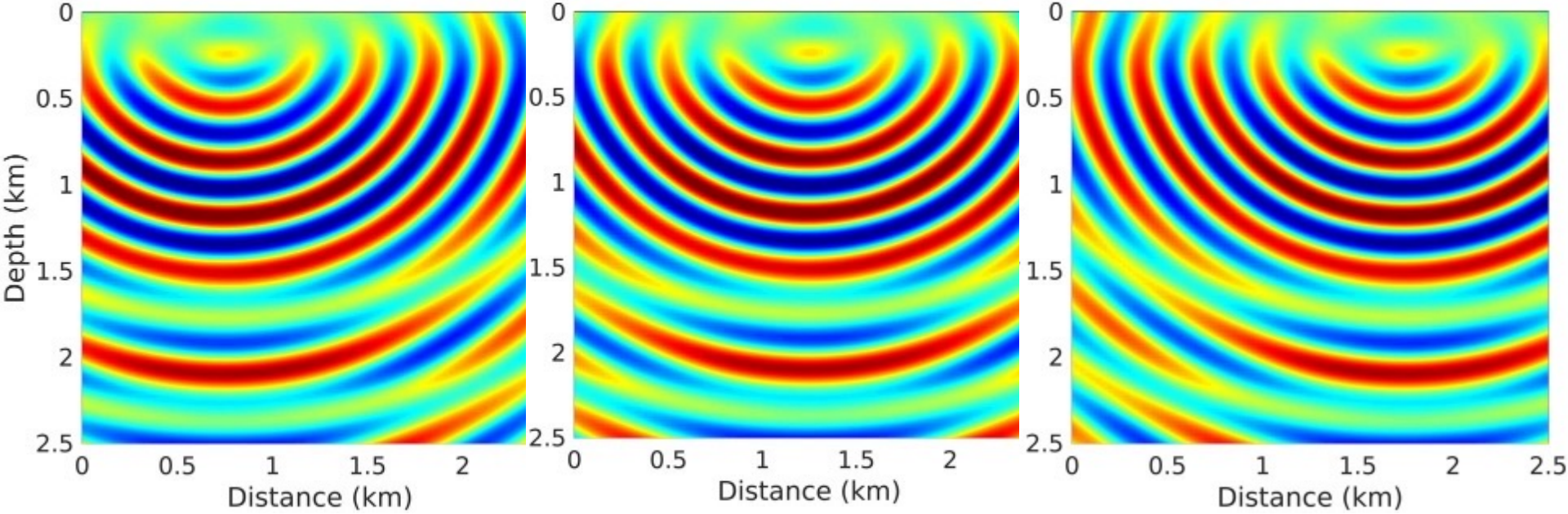


S_x=0.75 km

S_x=1.25 km

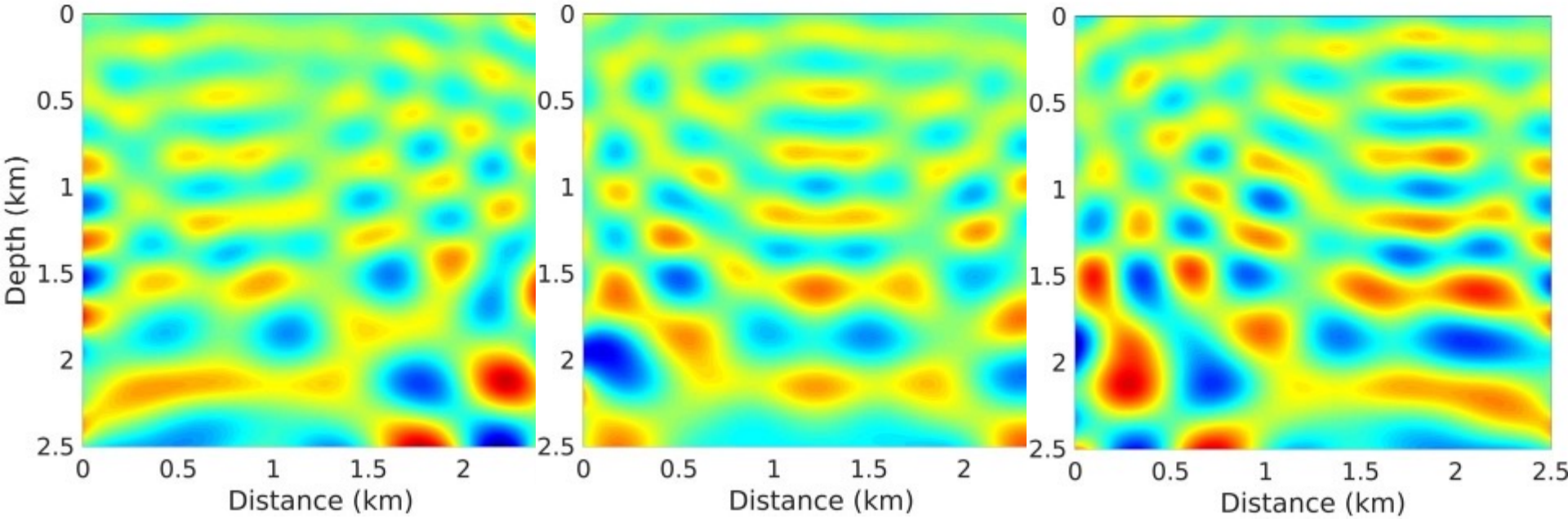
S_x=1.75 km

The NN predicted scattered wavefield (real part)



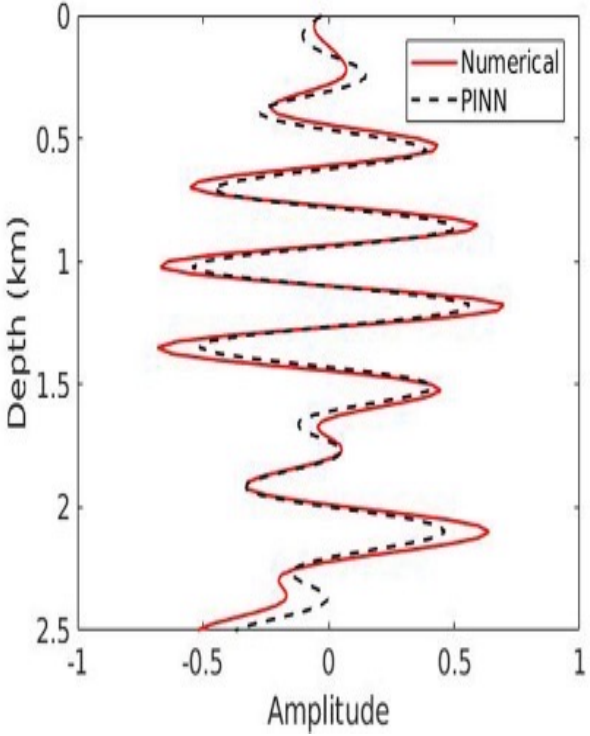
S_x=0.75 km ***S_x=1.25 km*** ***S_x=1.75 km***

The difference (real part)

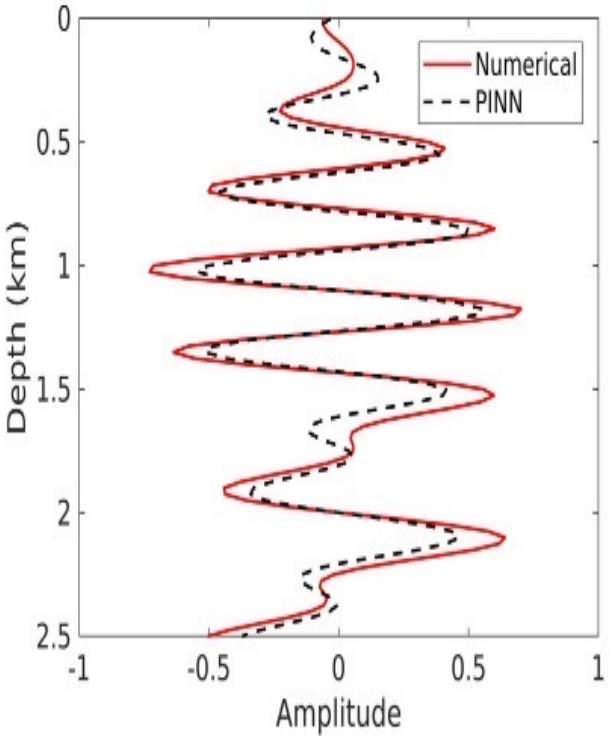


S_x=0.75 km ***S_x=1.25 km*** ***S_x=1.75 km***

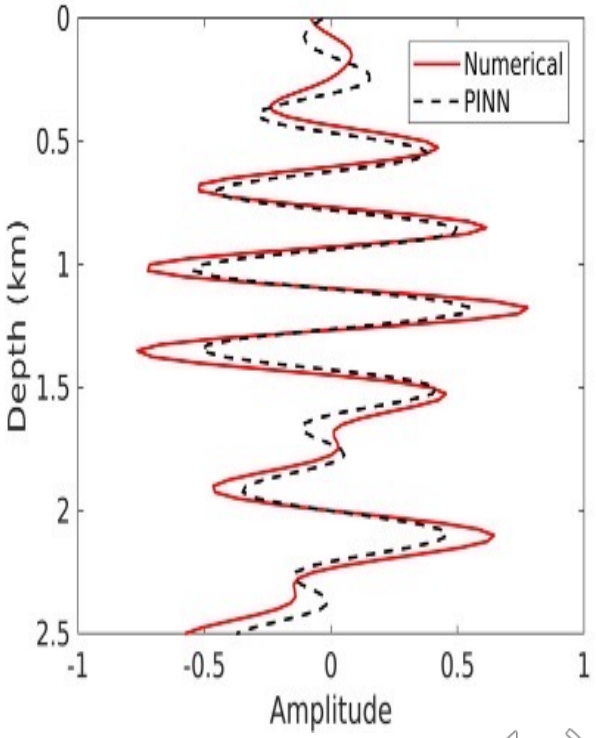
The difference (real part)



S_x=0.75 km



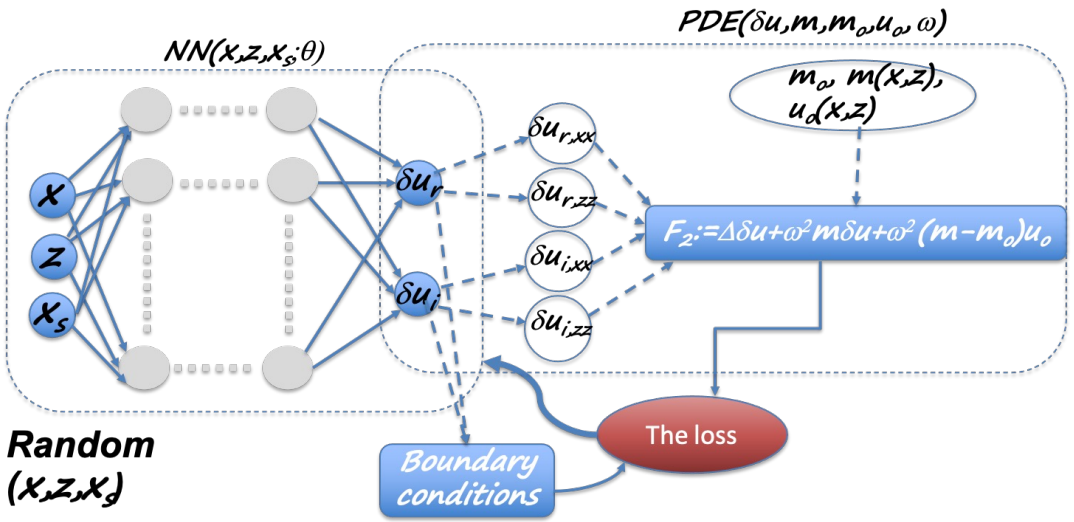
S_x=1.25 km

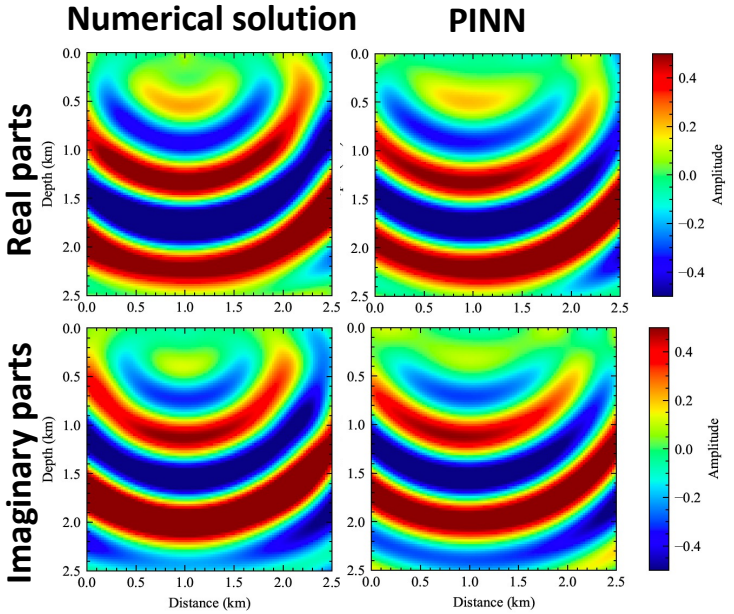


S_x=1.75 km

2Hz wavefields (x,z,sx), model: {4,4}, 96 param.

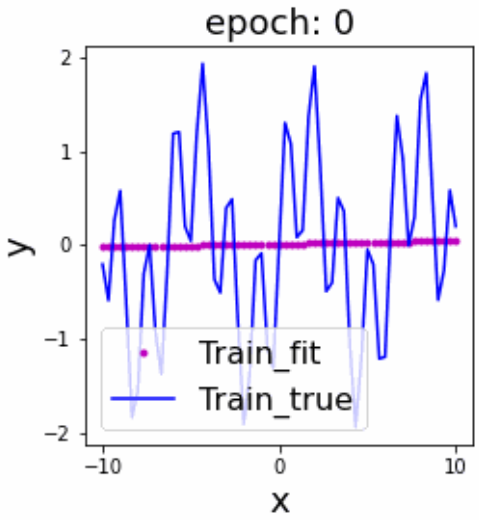
$$\boldsymbol{u} = \boldsymbol{f}(x_s, x, z;\ \boldsymbol{\theta})$$

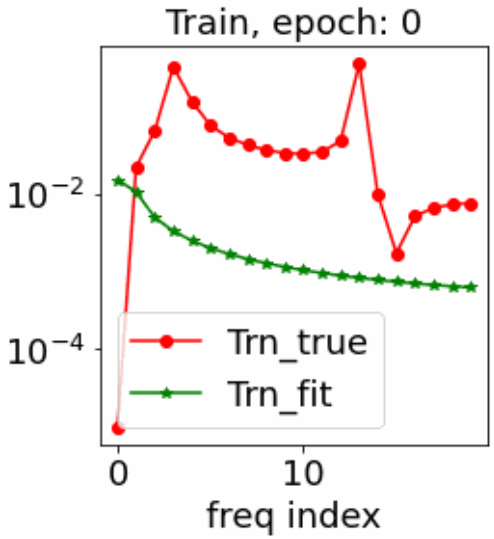




Low frequency Bias?

Low frequency bias



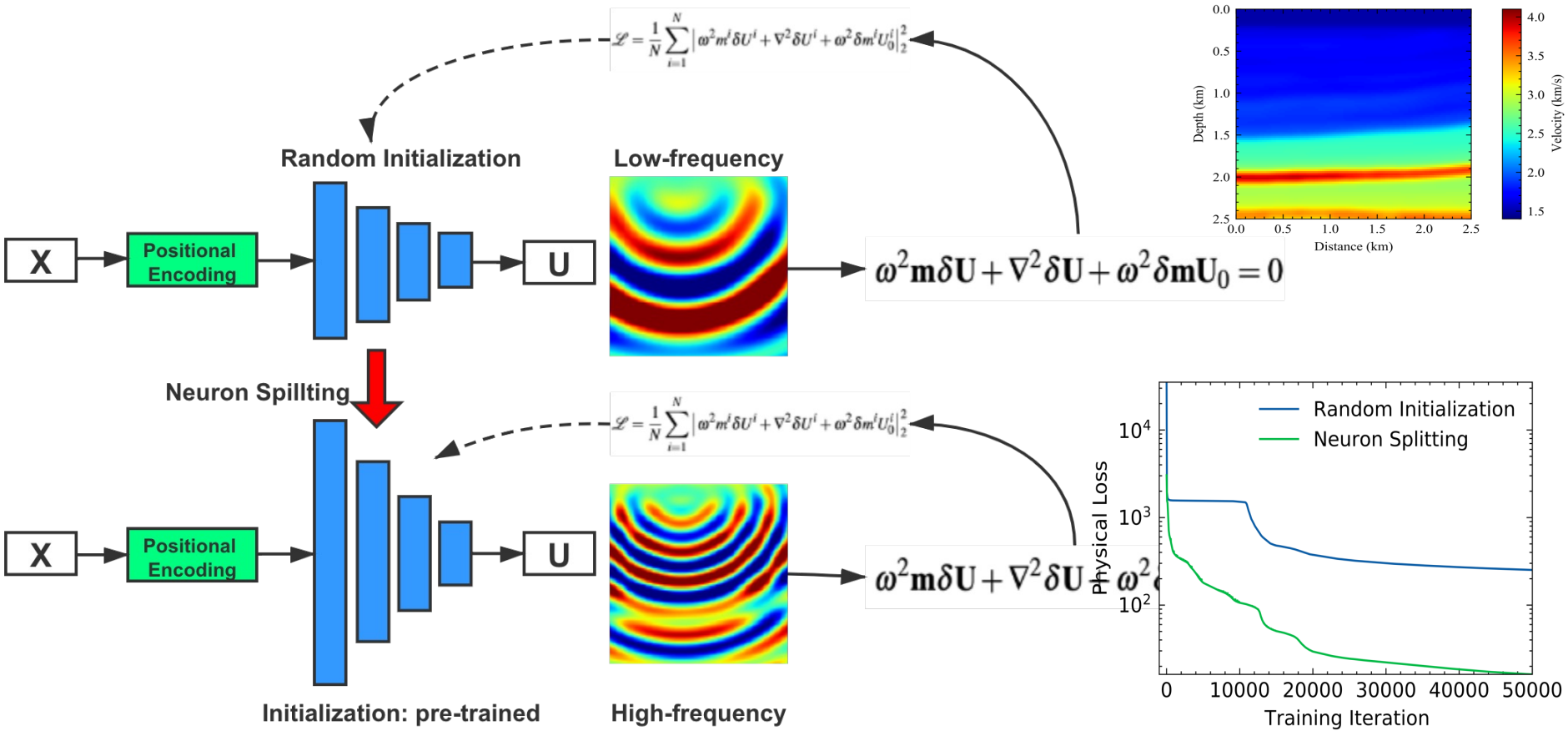


Jean-Baptiste Joseph Fourier (1768-1830)

PINNup

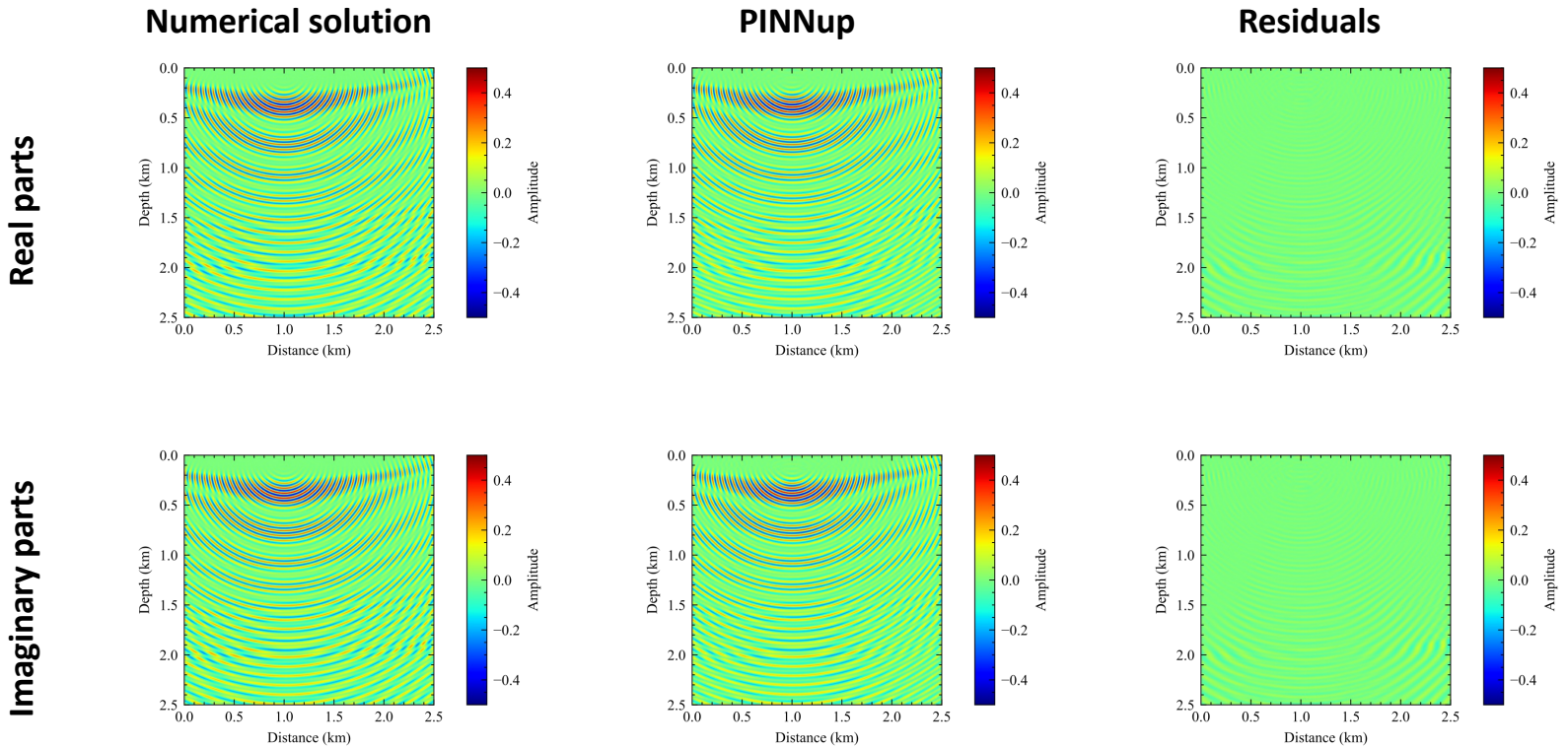
- **Low frequency bias** (N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, “On the spectral bias of neural networks,” arXiv, no. 1, 2018)
- **Low frequency to high**, in the sprit of FWI
- **Neuron Splitting** (Q. Liu, L. Wu, and D. Wang, “Splitting steepest descent for growing neural architectures,” arXiv preprint arXiv:1910.02366, 2019)

Frequency upscaling and Neoron splitting



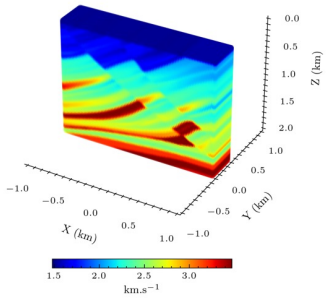
Huang and Alkhalifah, 2021, PINNup: Robust neural network wavefield solutions using frequency upscaling and neuron splitting, arXiv.

3D wavefields (x, z, x_s, 32 Hz) {1024,1024}

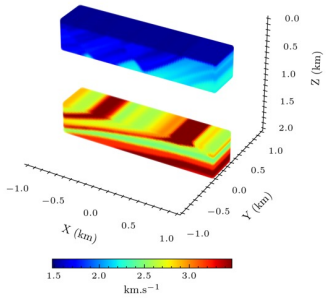


PINN's interpolation properties

Sliced Original* Velocity



Input to PINNs

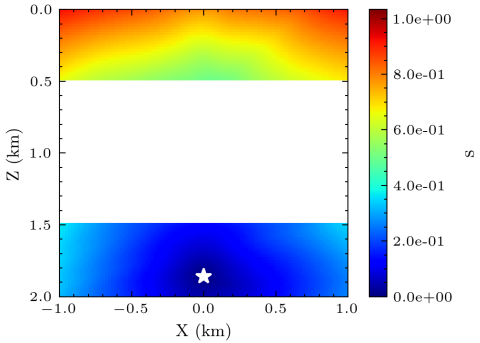


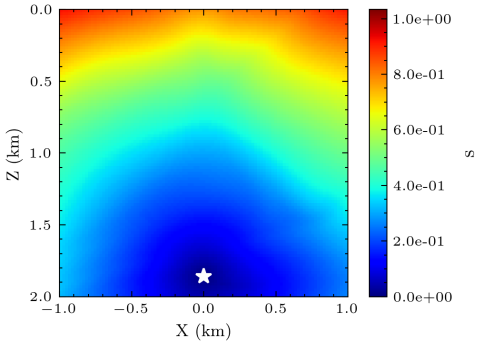
50% missing velocity

Input (with gaps)

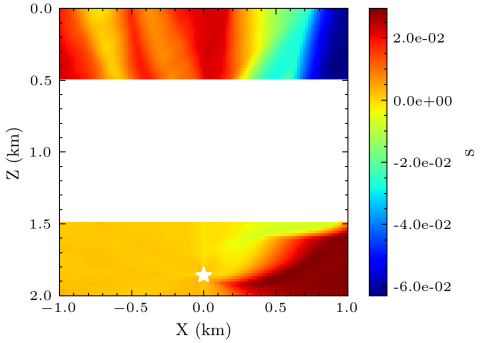
Original Domain

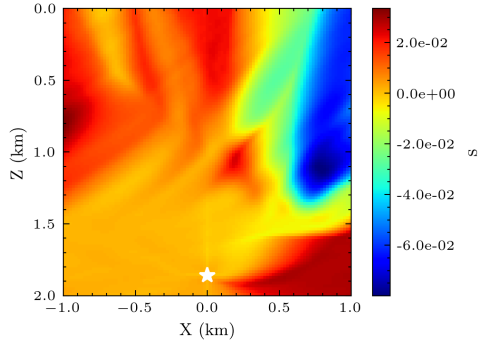
PINNs Calculation





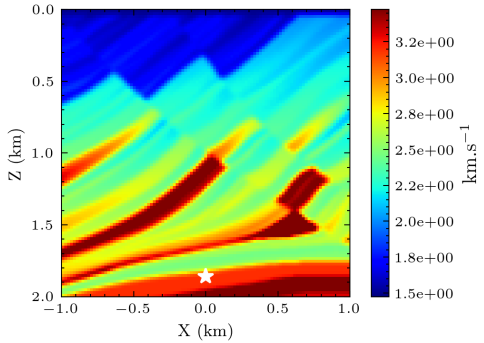
Traveltime error



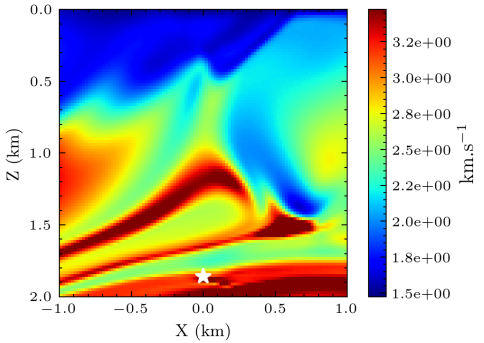


The velocity model

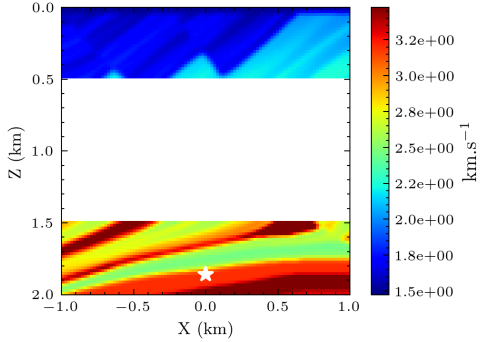
Original Velocity



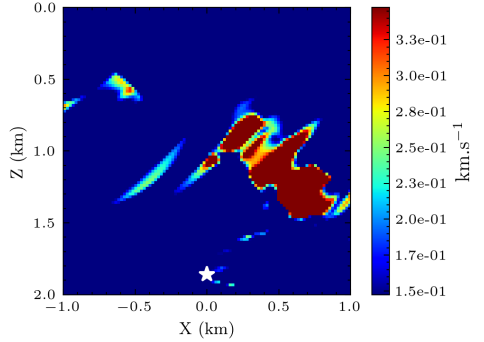
PINNs Output



Input Velocity

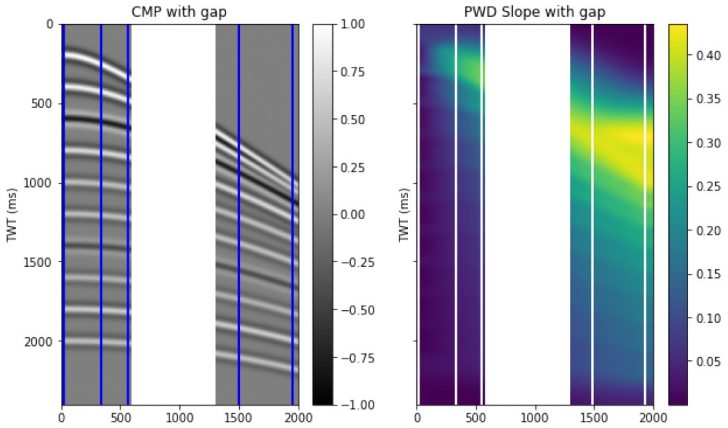


Relative Error



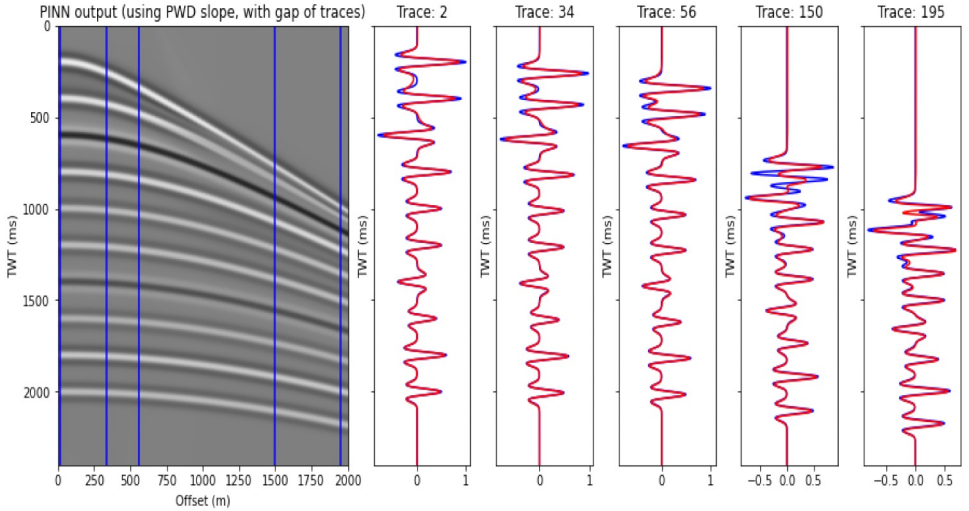
50% missing velocity

PINNs for geophysical applications – Seismic interpolation

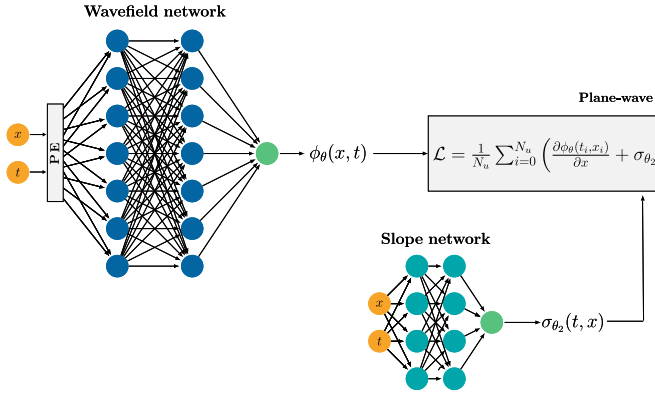


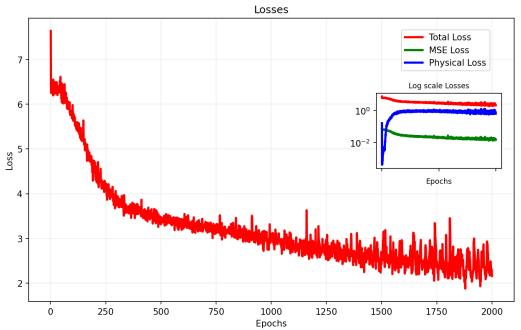
Network: $u = f(x, t; \theta)$

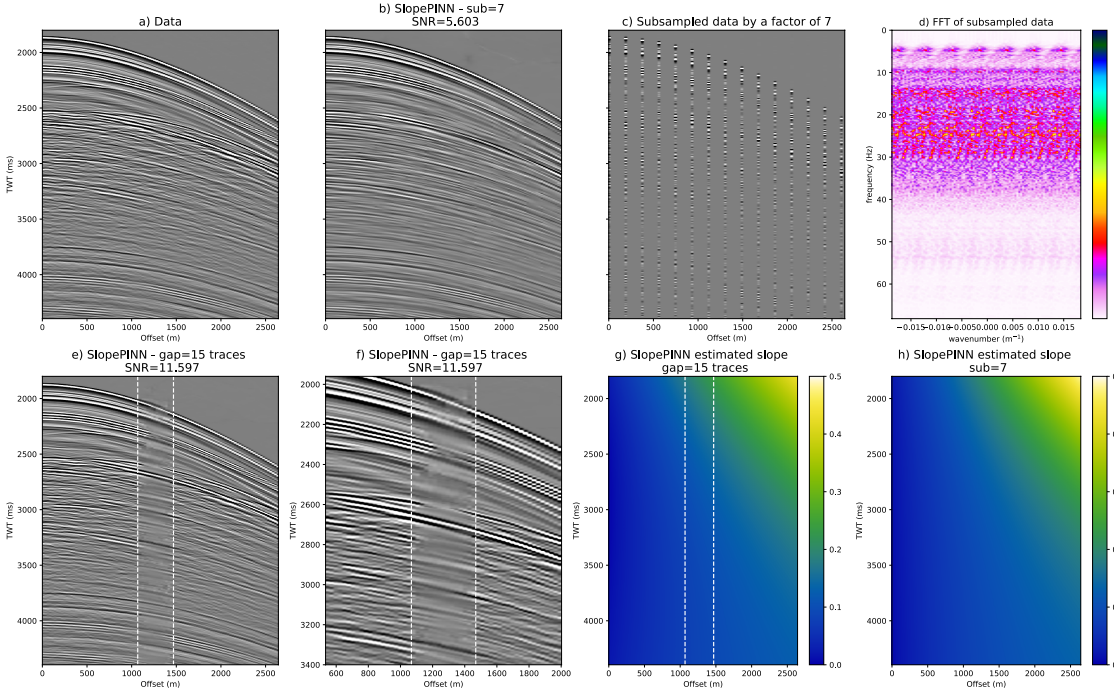
$$PDE_u = u_x + \sigma u_t$$



SlopePINN: physics-informed neural network slope estimation and interpolation with positional encoding







Imaging

The governing equation:

$$\frac{\omega^2}{v^2(x,z)} u(x,z,\omega) + \nabla^2 u(x,z,\omega) = 0$$

subject to the boundary condition:

$$u(x,z,\omega) = D(x,\omega)$$

The time=0 imaging condition:

$$I(x,z) = \int u(x,z,\omega)\, d\omega$$

PINN imaging

The conventional PINN loss:

$$L = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{\omega^2}{v^2} u^i + \nabla^2 u^i \right| + a \frac{1}{N_b} \sum_{i=1}^{N_b} |D^i - u^i|$$

Over random (x_i, z_i, ω_i) for $i = 1, N$,
and over recorded (x_i, ω_i) for $i = 1, N_b$

Hard constraints

To inherently include the boundary condition:

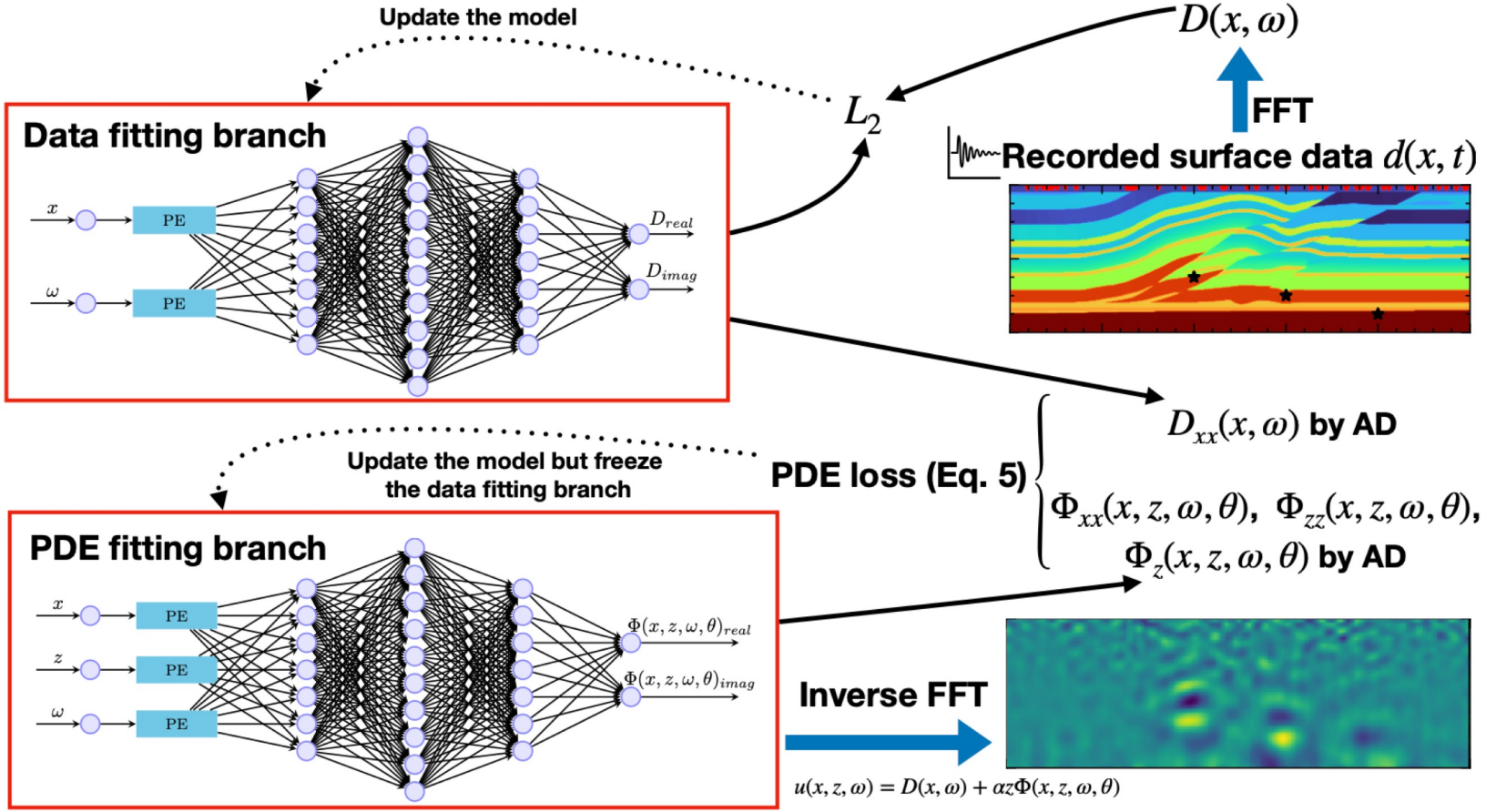
$$u(x, z, \omega) = D(x, \omega) + z\Phi(x, z, \omega, \theta),$$

The corresponding Loss:

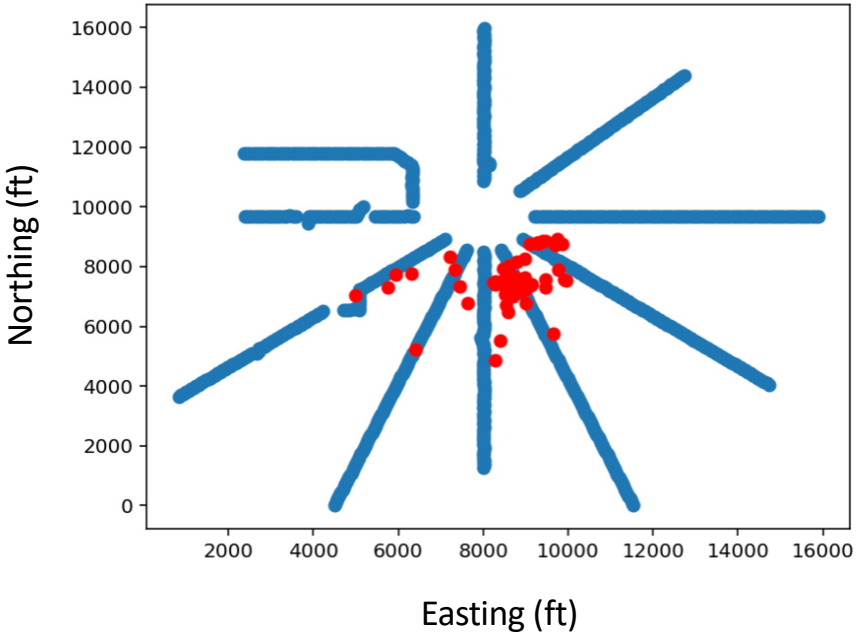
$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{z}^i\frac{(\omega^i)^2}{(v^i)^2}\Phi(\mathbf{x}^i, \mathbf{z}^i, \omega^i, \theta) + \mathbf{z}^i\nabla^2\Phi(\mathbf{x}^i, \mathbf{z}^i, \omega^i, \theta) + \right.$$

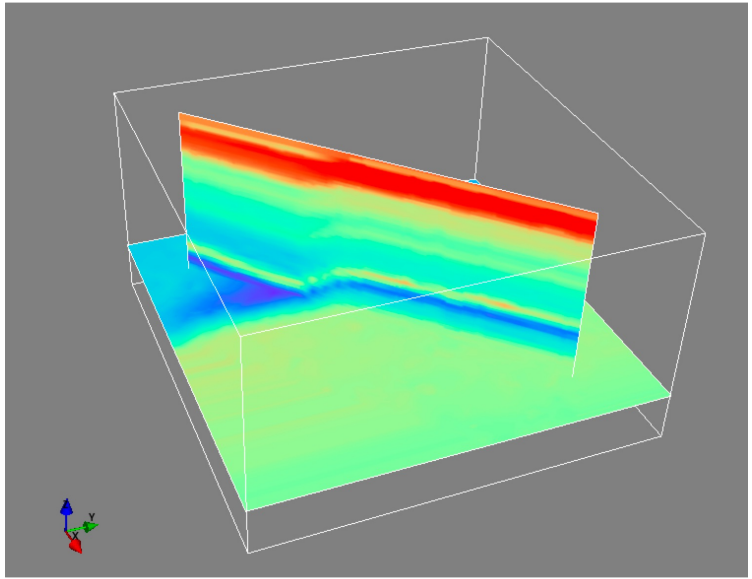
$$\left. 2\nabla_z\Phi(\mathbf{x}^i, \mathbf{z}^i, \omega^i, \theta) + \frac{(\omega^i)^2}{(v^i)^2}D(\mathbf{x}^i, \omega^i) + 2\nabla_x^2 D(\mathbf{x}^i, \omega^i)\right|_2^2$$

Application on source imaging

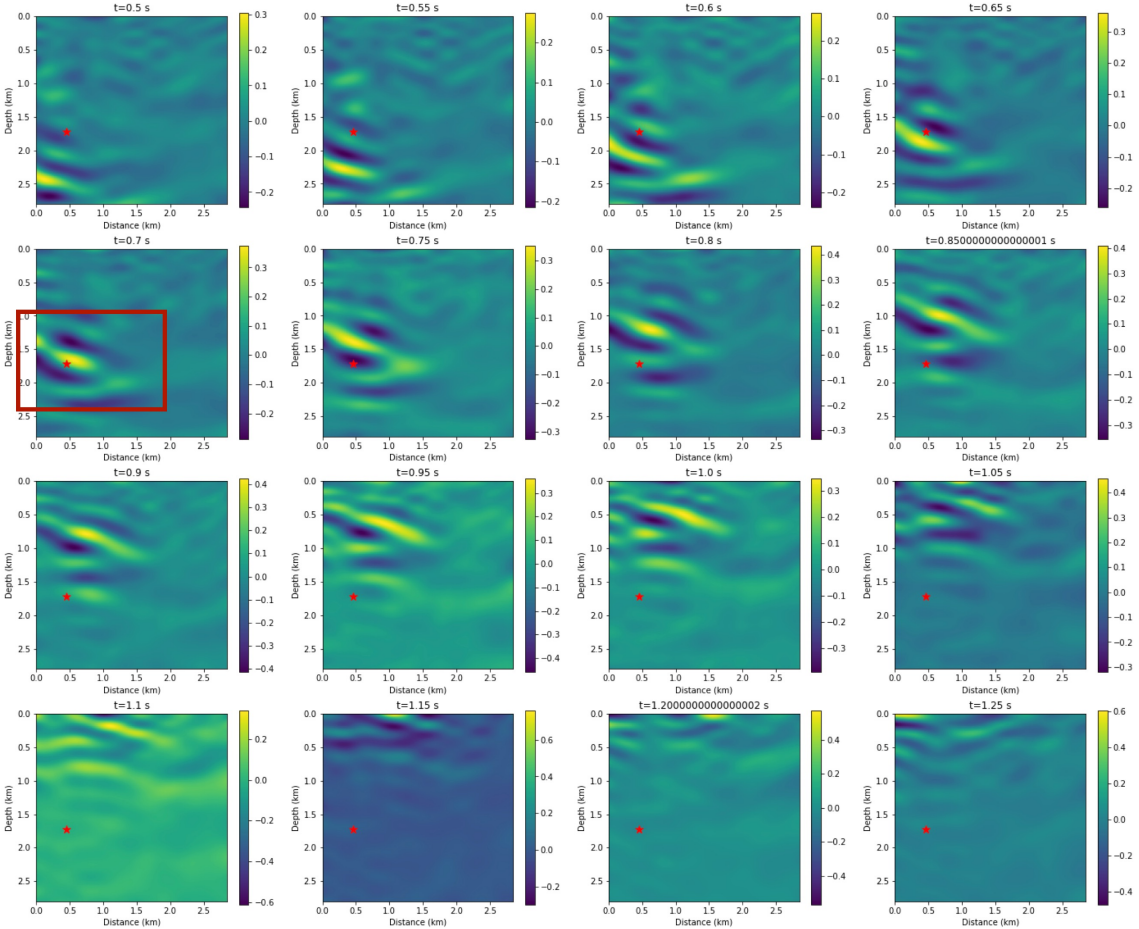


Field tests: (Oklahoma Arkoma Basin)





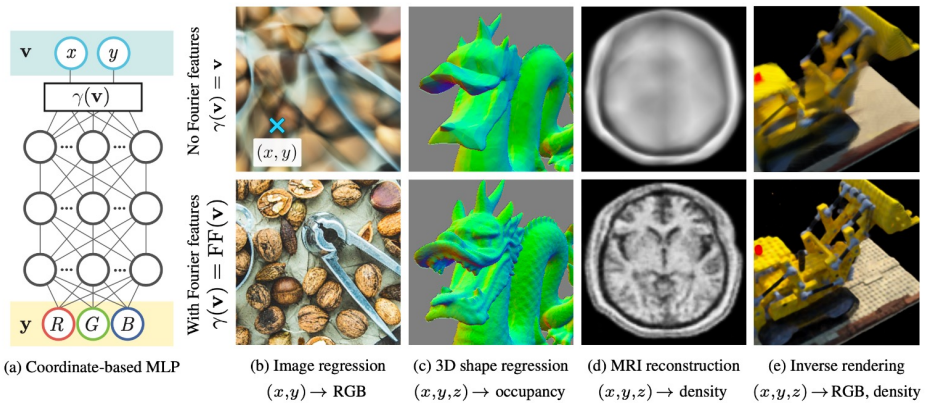
Snapshots in the time domain

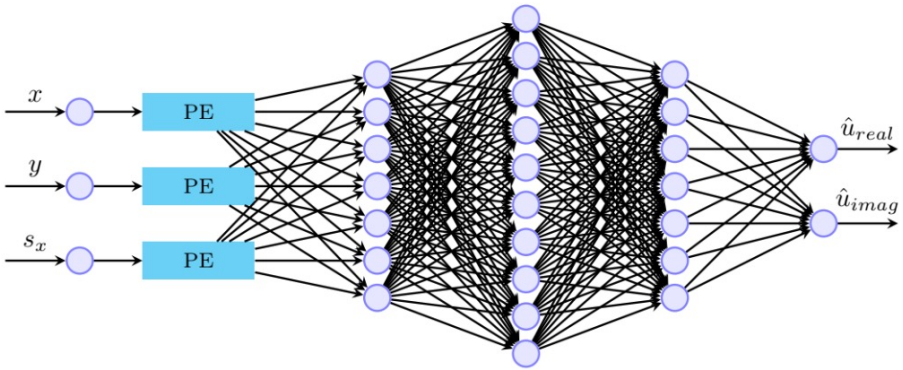


Cost? *The elephant in the room.*

Encoding

- High dimensional represtation of the scalar input.
- More weight to the location within the network (point source versus plane source).
- Positional encoding, barrowed from Transformers (Fourier encoding in NeRF).

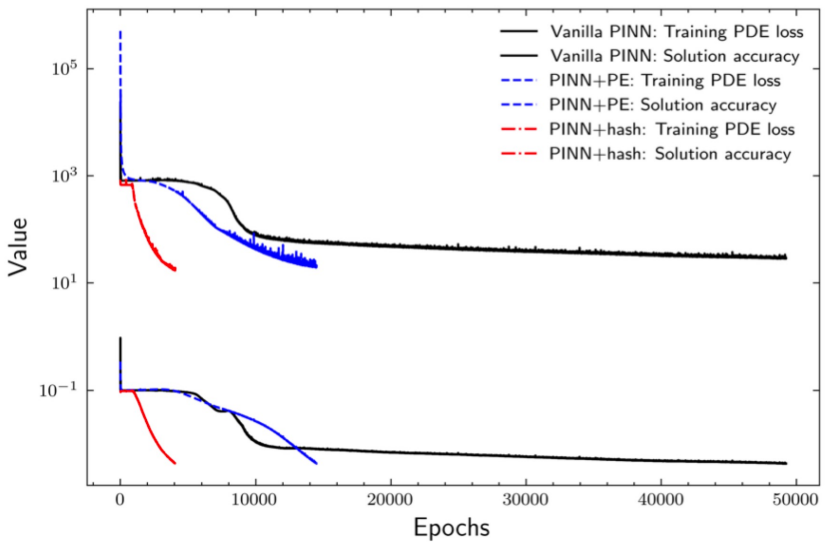


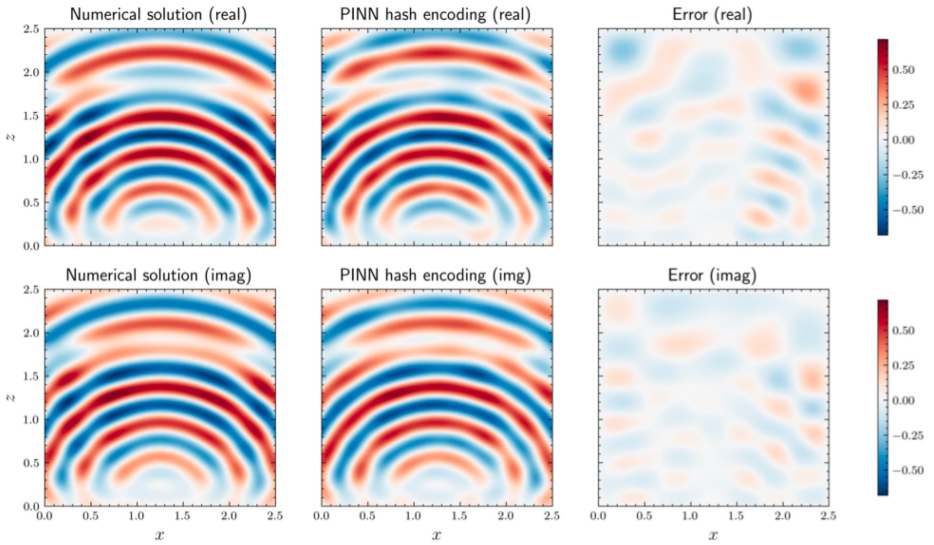


Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil,S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; and Ng, R. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems, 33.

Helmholtz equation (scattered wavefield)

Reduced from 491 s to 39 s

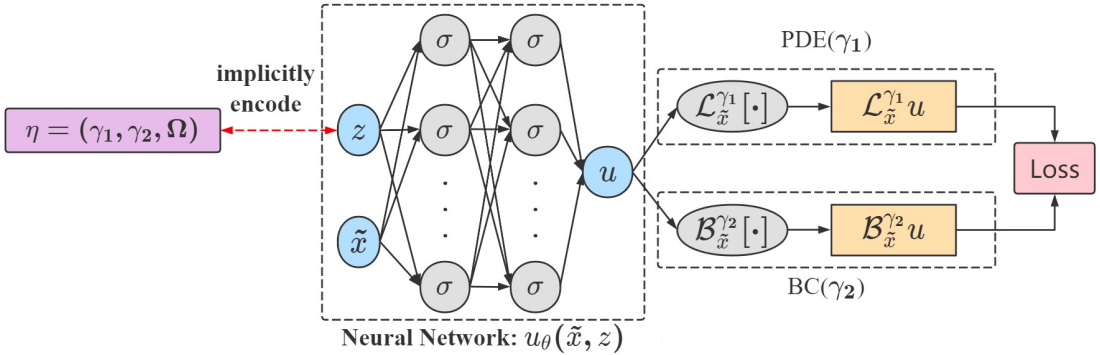




T=8, MLP: {144,144,144}, MLP with hash encoding: {128,128,128}

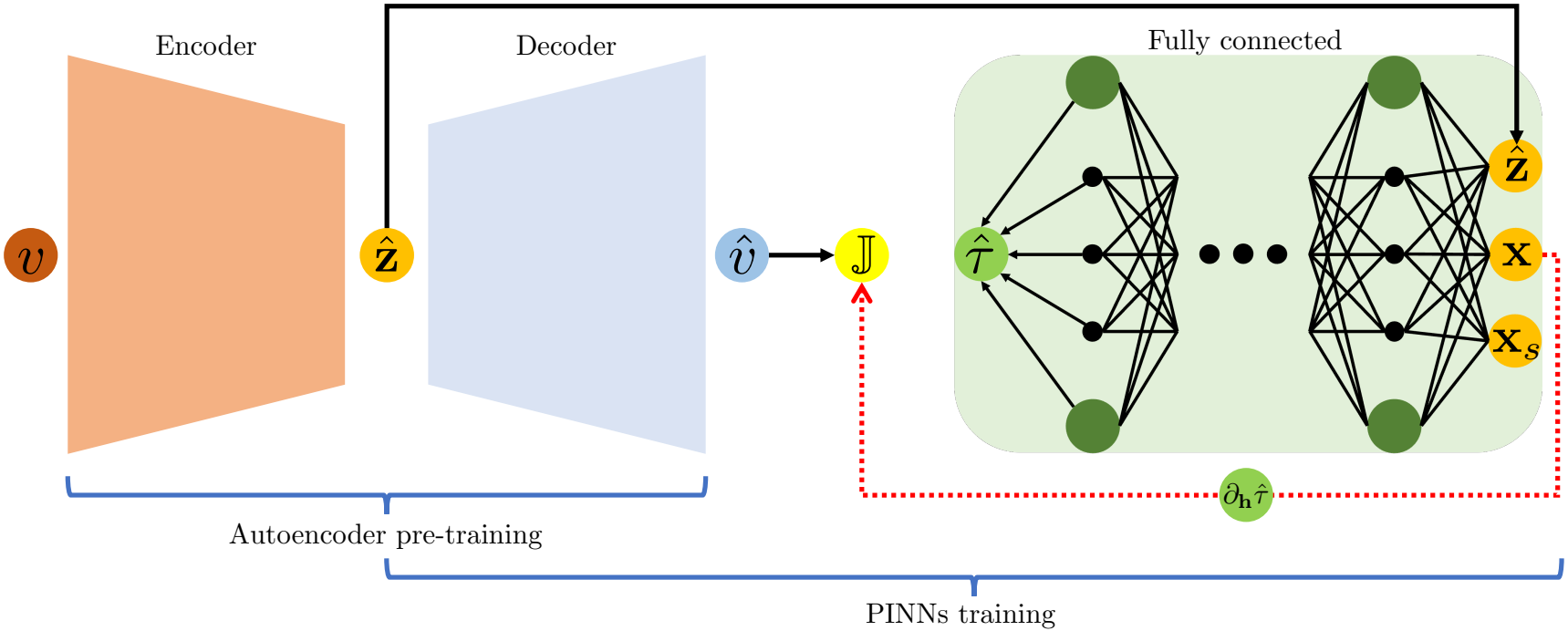
Generalization

- -Solutions for a specific PDE parameter
- -Transfer learning?
- -Meta-PDE→ learning a good initialization
- -Meta-Auto-Decoder

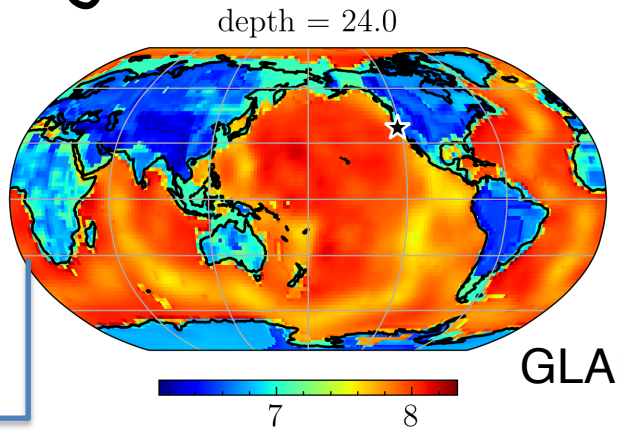


Huang X., 2023: Meta-Auto-Decoder for Solving Parametric Partial Differential Equations, arXiv

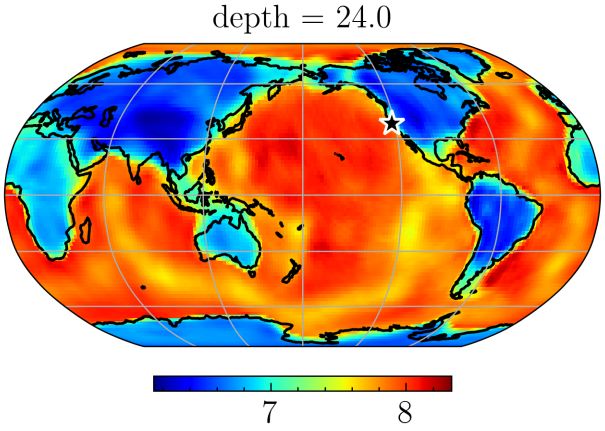
LatentPINN: a generator PINN

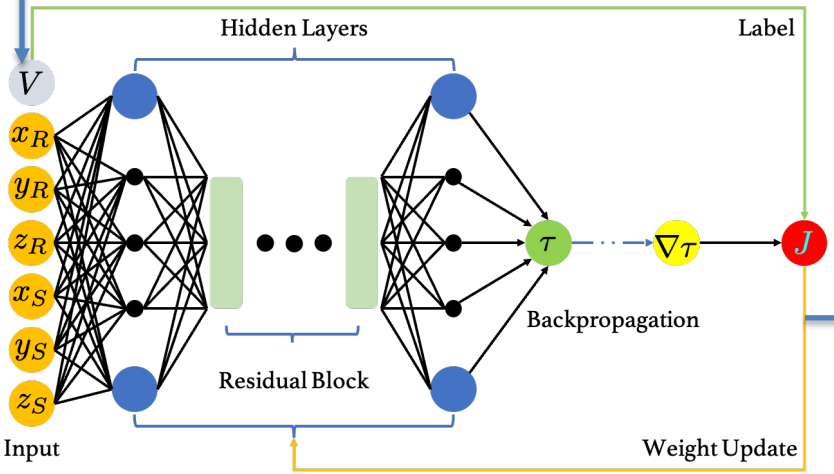


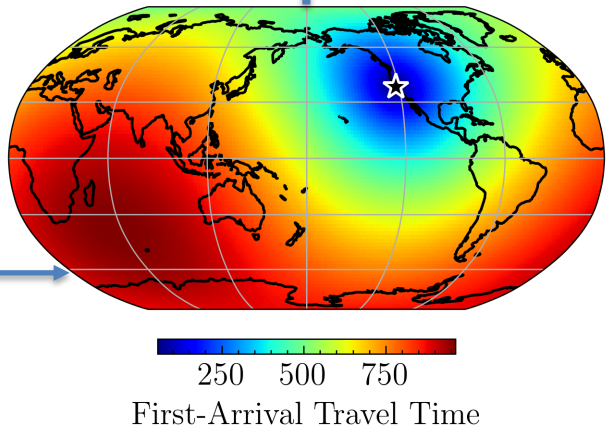
Embedding the Earth velocity in an NN



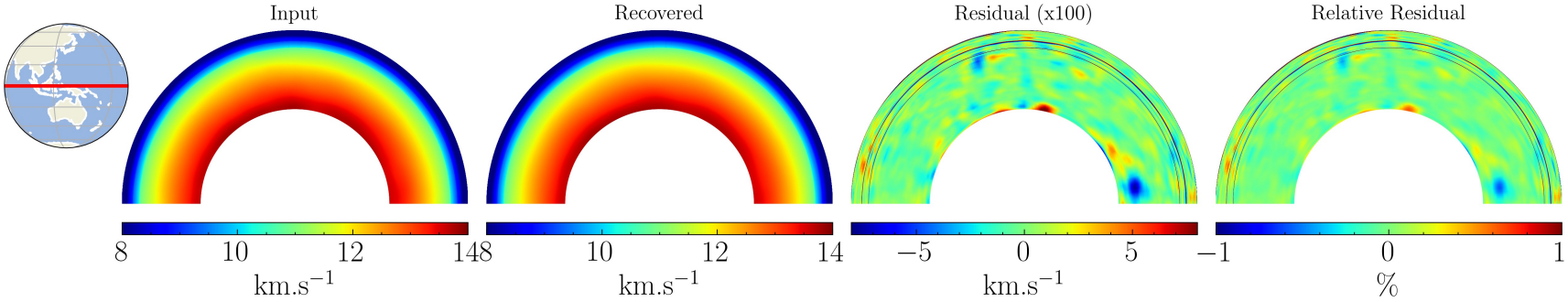
GLAD-M25 model

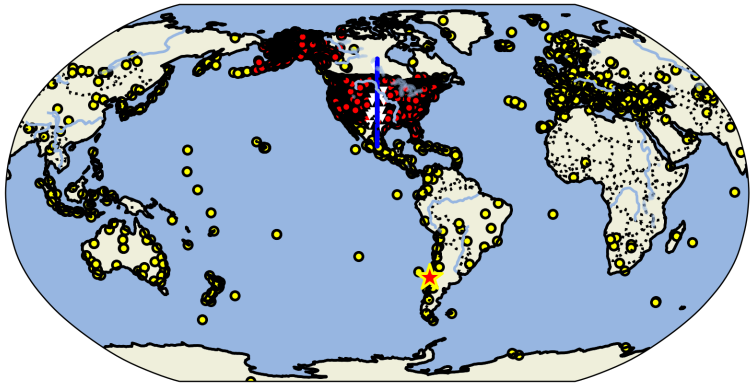




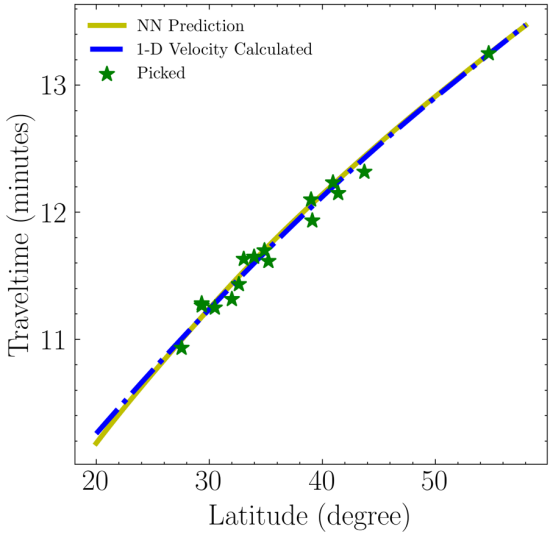


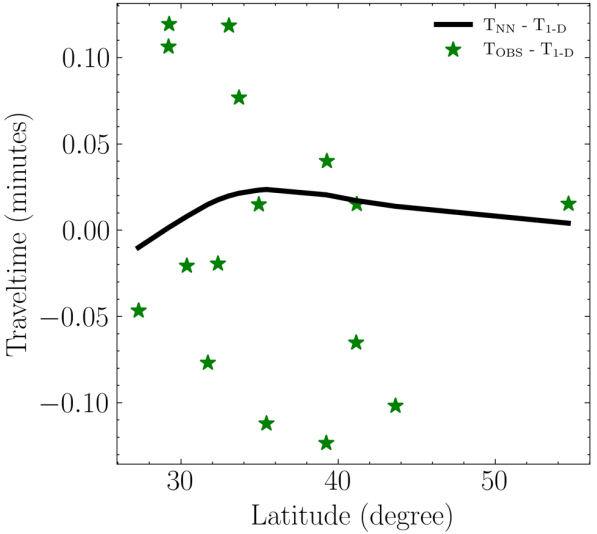
Accuracy and interpolation power





Earthquake in Chile

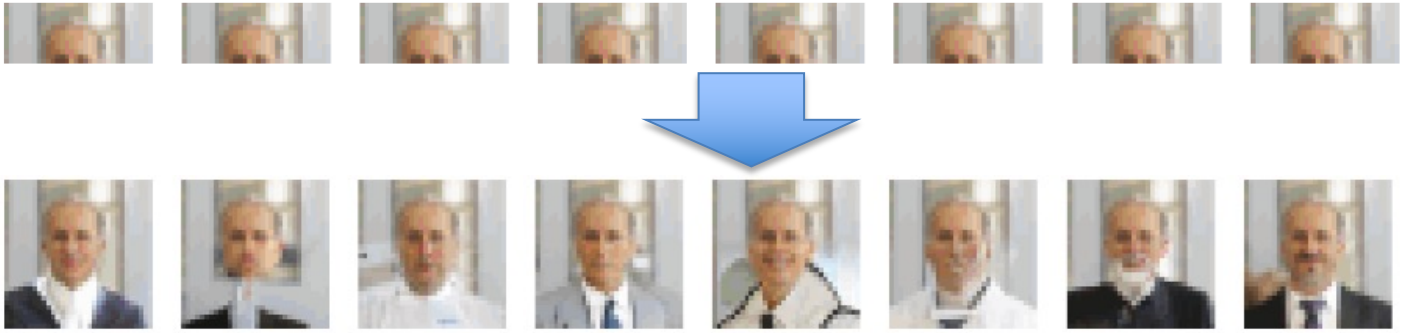




Key points

- *PINNs are nice, but challenges exist*
- *Positional encoding, PINNup, and many other*
- *Interpolation properties*
- *Hash encoding → fast convergence*
- *latentPINN → generalization*

Acknowledgements



- My group who inspire me all the time
- We thank KAUST for its continuous support of our work
- Thank you for being here


Thank You
30 17:50

From Conservation Laws

- Continuity Equation (Mass Conservation):

$\partial \rho / \partial t + \nabla \cdot (\rho \mathbf{v}) = 0$

- ρ is the density, $\mathbf{v}$ is the velocity field.

- Momentum Conservation (Euler's Equation):

$\rho \, d\mathbf{v}/dt = - \nabla p$

- p is the pressure field.

- Linearization Assumption:

- Assume small perturbations around equilibrium state ρ_o, p_o.

The Wave Equation

- Combining continuity and momentum equations, assuming a homogeneous medium:

  $\partial^2 p/\partial t^2 = c^2 \nabla^2 p$

  - $c = \mathrm{sqrt}(1/\rho_0 * dp/d\rho)$ is the speed of sound.
- This equation describes pressure perturbations propagating as waves.
- Applications: Acoustics, medical ultrasound, noise control.

Neural Tangent Kernel (NTK)

- The NTK describes training dynamics:

$$K(\mathbf{x}, \mathbf{x}') = \sum_i \partial u_\theta(\mathbf{x},t)/\partial \theta_i \, \partial u_\theta(\mathbf{x}',t)/\partial \theta_i$$

- In the infinite-width limit, training becomes linear.
- Helps analyze convergence of PINNs.

$$\partial u_\theta(\mathbf{x},t)/\partial t = -\eta \, K(\mathbf{x}, \mathbf{x}') \, \partial L/\partial u_\theta(\mathbf{x},t)$$

Applications of NTK in PINNs

- Accelerating training by optimizing network initialization
- Understanding loss landscape and convergence rates
- Improving generalization of PINNs by leveraging NTK properties
- Developing adaptive architectures for better physics enforcement