SD-PINN: Physics informed neural networks for spatially dependent PDEs

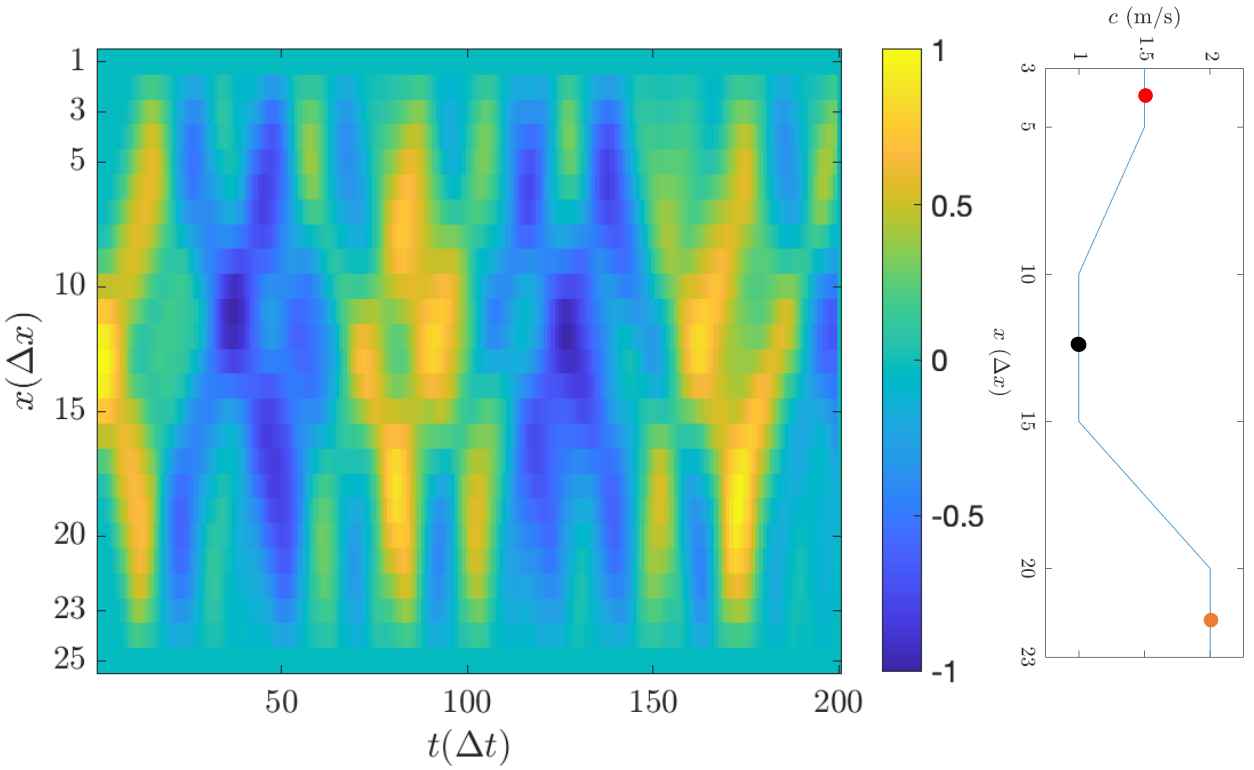
Ruixian Liu, Peter Gerstoft

University of California, San Diego

Spatially dependent PDEs

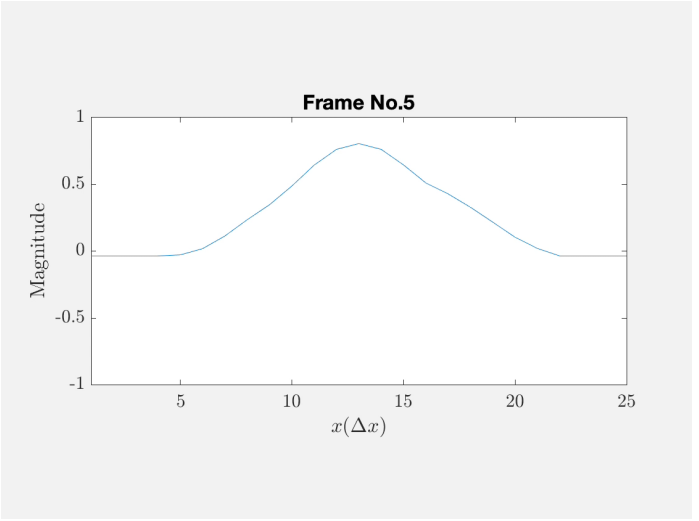
Wave equation without attenuating at M locations:

$$U_{tt} = c_m^2 U_{xx},\ m = 1, ..., M\ .$$



The PDE (wave equation) is spatially dependent at M locations due to various phase speeds:

- $U_{tt} = 2.25 U_{xx},\ c = 1.5$ m/s
- $U_{tt} = U_{xx},\ c = 1$ m/s
- $U_{tt} = 4 U_{xx},\ c = 2$ m/s

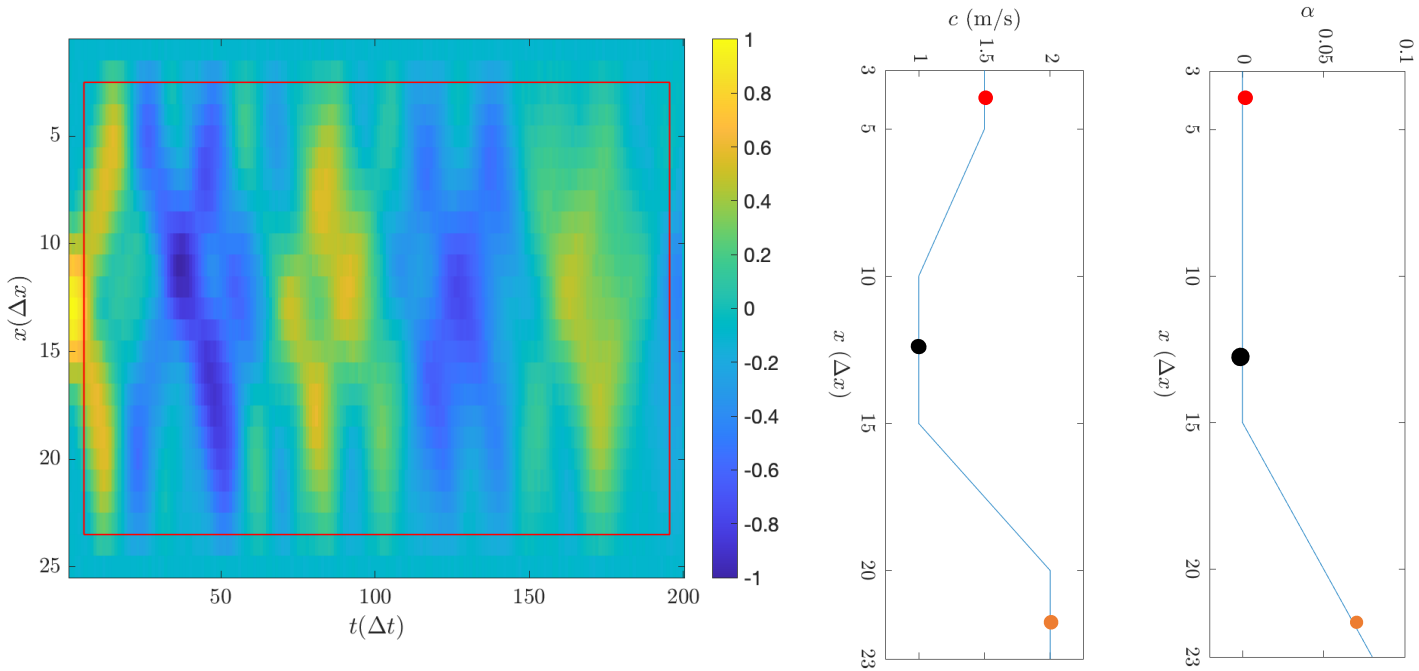


Spatially dependent PDEs

Wave equation with attenuating at M locations:

$$U_{tt} = -\alpha_m U_t + c_m^2 U_{xx}$$

$$m = 1, ..., M\ .$$



The PDE (wave equation) is spatially dependent at M locations due to various phase speeds and attenuation factors:

- $U_{tt} = 2.25 U_{xx},\ c = 1.5\ \text{m/s},\ \alpha = 0$
- $U_{tt} = U_{xx},\ c = 1\ \text{m/s},\ \alpha = 0$
- $U_{tt} = -0.07 U_t + 4 U_{xx},\ c = 2\ \text{m/s},\ \alpha = 0.07$

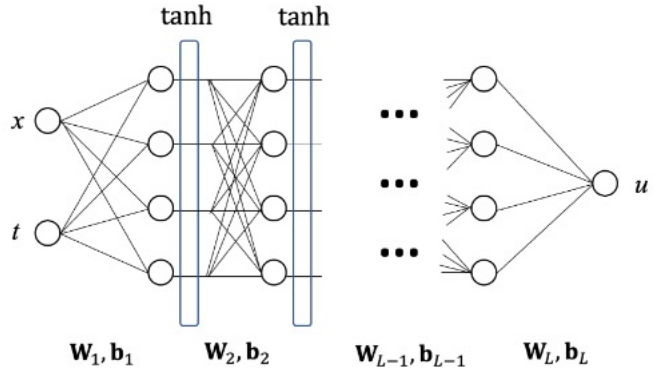
For the 1-dimensional single frequency wave solution:

$$U = U_0 e^{-\alpha' x} \text{Re}(e^{i(kx - \omega t)})$$

the attenuating wave equation links the attenuation factor in the PDE α with the attenuation in the solution α' in Np/m by:

$$\alpha' = \sqrt{\frac{\omega}{2c^2}(\sqrt{\alpha^2 + \omega^2} - \omega)}$$

Network structure – Fully connected neural network (FCN)



The network is parameterized by θ (wights and bias in all layers):

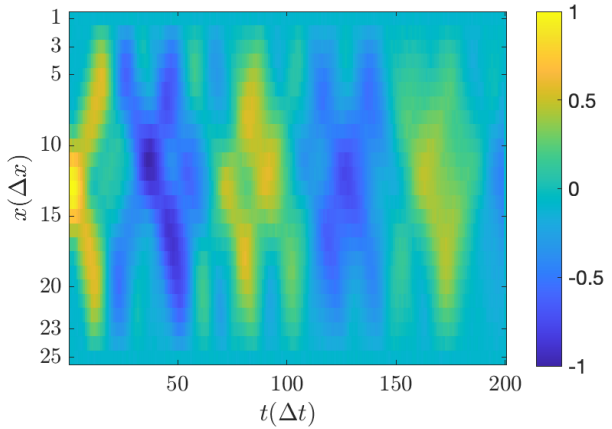
$$Net_\theta(x,t) = \mathbf{W}_L^{\mathrm{T}}(\dots(\mathbf{W}_2^{\mathrm{T}}\tanh(\mathbf{W}_1^{\mathrm{T}}\begin{bmatrix}x\\t\end{bmatrix}+\mathbf{b}_1)+\mathbf{b}_2)\dots)+\mathbf{b}_L$$

The PDE is parameterized by λ .

For the PDE: $\ell_m^j \quad \lambda_{m1} \; d_{m1}^j \quad \lambda_{m2} \; d_{m2}^j$

$$(U_{tt})_m^j = -\alpha_m(U_t)_m^j + c_m^2(U_{xx})_m^j \qquad \ell_m^j = \sum_{k=1}^{K}\lambda_{mk}d_{mk}^j$$

we have $\lambda = \begin{bmatrix}-\alpha_1 & \cdots & -\alpha_M \\ c_1^2 & \cdots & c_M^2\end{bmatrix}^{\mathrm{T}}$



Loss functions - Overall - $loss = loss_{\mathrm{u}} + w_{\mathrm{f}} \times loss_{\mathrm{f}} + w_{\mathrm{sm}} \times loss_{\mathrm{sm}} + w_{\mathrm{b}} \times loss_{\mathrm{b}} + w_{\mathrm{si}} \times loss_{\mathrm{si}}$

Data loss - $$loss_{\mathrm{u}}(\theta) = \sum_{m=1}^{M}\sum_{j=1}^{T}(\widehat{u}_{mj}(\theta) - u_{mj})^2$$

Functional loss - $$(U_{tt})_m^j = -\alpha_m(U_t)_m^j + c_m^2(U_{xx})_m^j$$

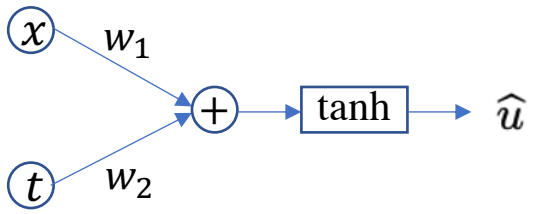
Estimations are computed by automatic differentiation:

$$(\widehat{U}_x)_m^j = \widehat{U}_x(x_m,t_j) = \left.\frac{\partial Net_\theta(x,t)}{\partial x}\right|_{x=x_m,t=t_j}$$

$$(\widehat{U}_{xx})_m^j = \widehat{U}_{xx}(x_m,t_j) = \left.\frac{\partial}{\partial x}\left(\frac{\partial Net_\theta(x,t)}{\partial x}\right)\right|_{x=x_m,t=t_j}$$

$$loss_{\mathrm{f}}(\lambda) = \sum_m\sum_j(\widehat{\ell}_m^j - (\sum_{k=1}^{K}\widehat{\lambda}_{mk}\widehat{d}_{mk}^j))^2 = \sum_m\sum_j((\widehat{U}_{tt})_m^j + \alpha_m(\widehat{U}_t)_m^j - c_m^2(\widehat{U}_{xx})_m^j)^2$$

An example for automatic differentiation:



$$\hat{u} = \tanh(\overbrace{w_1 x + w_2 t}^{A})$$

$$\begin{aligned}\hat{U}_x = \frac{\partial \hat{u}}{\partial x} &= \frac{\partial \tanh A}{\partial A} \cdot \frac{\partial A}{\partial x} \\ &= \left(1 - \tanh^2 A\right) w_1\end{aligned}$$

$$\begin{aligned}\hat{U}_{xx} = \frac{\partial \left(\frac{\partial \hat{u}}{\partial x}\right)}{\partial x} &= \frac{\partial \left(\left(1 - \tanh^2 A\right) w_1\right)}{\partial x} \\ &= w_1 \cdot \frac{\partial (1 - \tanh^2 A)}{\partial x} \\ &= w_1 \cdot \frac{-\partial \tanh^2 A}{\partial x} \\ &= -w_1 \cdot \frac{\partial \tanh^2 A}{\partial \tanh A} \cdot \frac{\partial \tanh A}{\partial A} \cdot \frac{\partial A}{\partial x} \\ &= -2 w_1^2 \tanh A (1 - \tanh^2 A)\end{aligned}$$

Advantage on noise-robustness:

The automatic differentiation is an analytical computation parameterized by neural network parameters (in this example, w_1 and w_2). The neural network parameters are updated not only to make $\hat{u}$ simulate noisy measurements, but also to ensure that the physical laws are obeyed (via $loss_f$). The $loss_f$ encourages the network parameters to ignore the influence from measurement noise, which does not obey the assumed PDE.

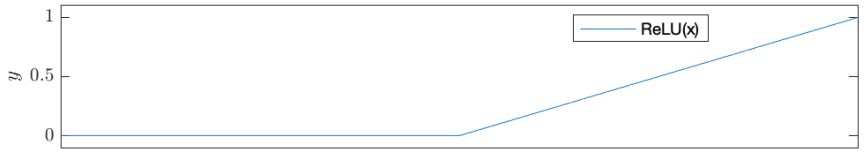
Sign loss-

We incorporate the knowledge of sign information.
E.g., for

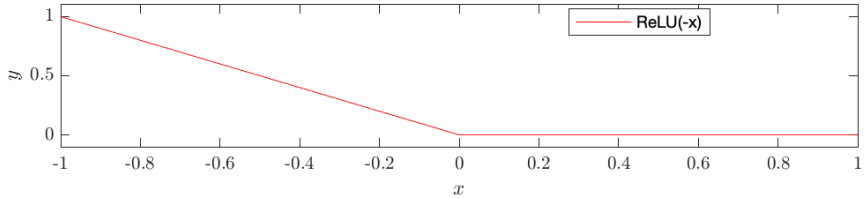
$$(U_{tt})_m^j = -\alpha_m (U_t)_m^j + c_m^2 (U_{xx})_m^j$$

the coefficient $c^2 \geq 0$ must hold since c is for the phase speed of the wave,
and $-\alpha \leq 0$ for the attenuation must hold for a system without input external energy.

$$loss_{\mathrm{si}}(\lambda) = \sum_m \sum_k \mathrm{ReLU}(-\mathrm{sign}(\lambda_{mk})\hat{\lambda}_{mk})$$



For here $loss_{\mathrm{si}} = \sum_m \mathrm{ReLU}(+(-\hat{\alpha}_m)) + \sum_m \mathrm{ReLU}(-(+\hat{c}_m^2))$



Smoothness loss-

$$loss_{\mathrm{sm}}(\lambda) = \sum_{k=1}^{K} \sum_{m=1}^{M-1} (\hat{\lambda}_{mk} + \hat{\lambda}_{(m+1)k} - 2\hat{\lambda}_{(m+0.5)k})^2$$

We recovers the PDE coefficients from both true positions and interpolated positions at the middle of 2 neighboring true positions.

Estimated coefficients at true positions: $\hat{\lambda}_m$ for integers m.
Estimated coefficients at interpolated places: $\hat{\lambda}_{m+0.5}$ for integers m.

The estimated coefficients at the middle of the 2 true locations should be near the average of the estimated coefficients at these true locations.

Boundary loss- We assume the PDE coefficients at spatial boundaries known, and use it to quickly obtain the estimated PDE coefficients at the boundaries by minimizing:

$$loss_{\mathrm{b}} = \sum_{k=1}^{K}((\widehat{\lambda}_{1k} - \lambda_{1k})^2 + (\widehat{\lambda}_{Mk} - \lambda_{Mk})^2)$$

Revised functional loss -

$$\begin{aligned} loss_{\mathrm{f}}(\lambda) &= \sum_{m\in I_{\mathrm{m}}}\sum_{j\in I_{\mathrm{t}}}(\widehat{\ell}_m^j - (\sum_{k=1}^{K}\widehat{\lambda}_{mk}\widehat{d}_{mk}^j))^2 \\ &= \sum_{m\in I_{\mathrm{m}}}\sum_{j\in I_{\mathrm{t}}}((\widehat{U}_{tt})_m^j + \alpha_m(\widehat{U}_t)_m^j - c_m^2(\widehat{U}_{xx})_m^j)^2 \end{aligned}$$

for $(U_{tt})_m^j = -\alpha_m(U_t)_m^j + c_m^2(U_{xx})_m^j$

which incorporates the interpolated points.

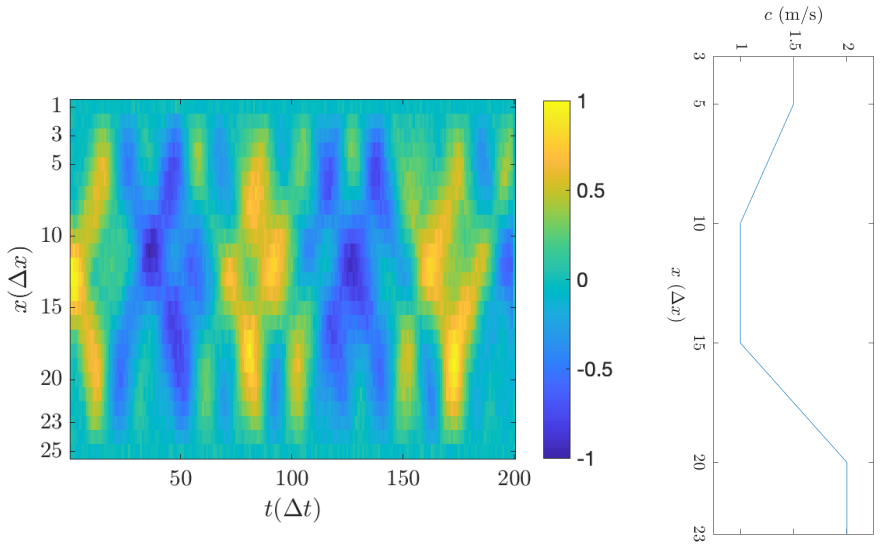
Datasets:

$\Delta t = 1$ s $\quad \Delta x = 2$ m $\quad$ Region of interest (ROI): $\quad [3, 23]\Delta x \quad [5, 195]\Delta t$

Non-attenuating waves

$$U_{tt} = c_m^2 U_{xx}$$

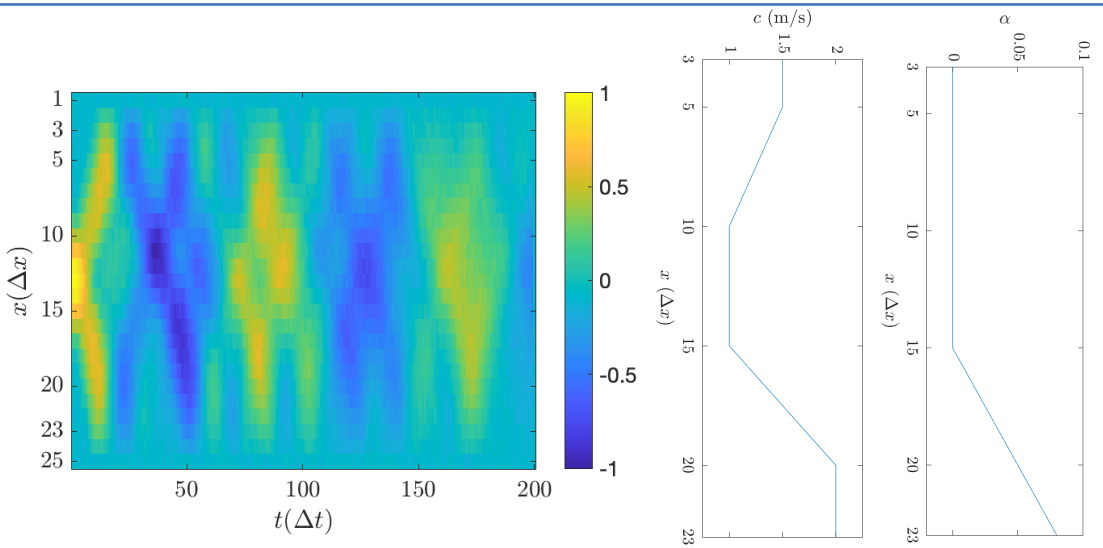
SNR = 20 dB



Attenuating waves

$$U_{tt} = -\alpha_m U_t + c_m^2 U_{xx}$$

SNR = 30 dB



Configurations:

Use Adam optimizer.

$w_{\mathrm{f}} = w_{\mathrm{sm}} = w_{\mathrm{b}} = w_{\mathrm{si}} = 10$

FCN has 9 layers.

Equally interpolate 3 points between 2 neighboring true time steps.

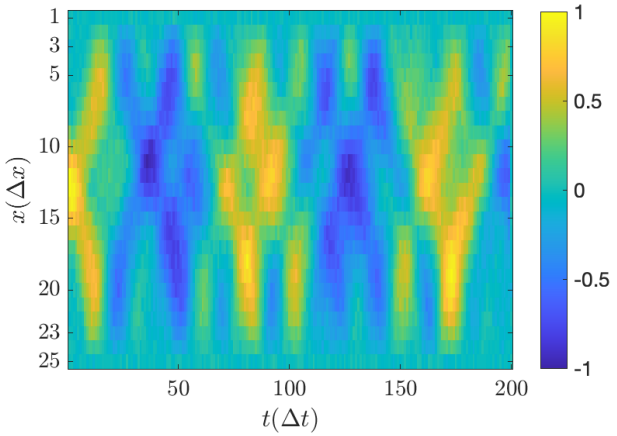
FCN has 5 layers.

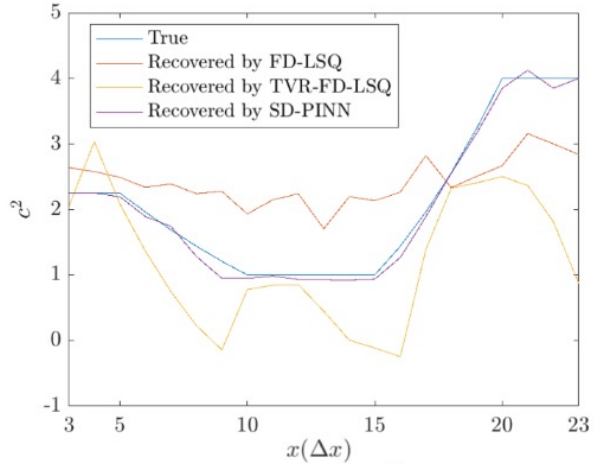
Interpolate 1 point at the middle of 2 neighboring true time steps.

Results

$$U_{tt} = c_m^2 U_{xx}$$

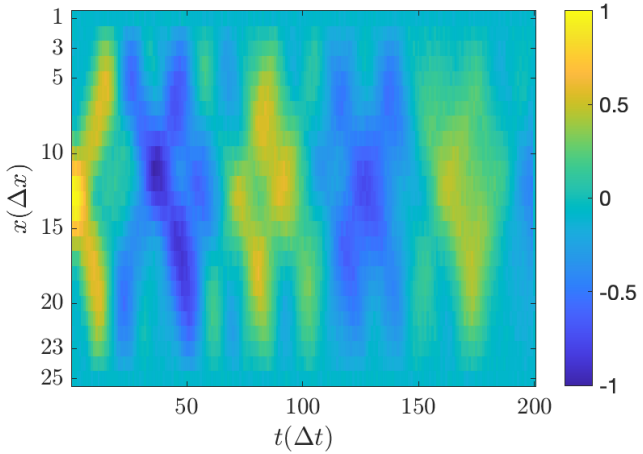
SNR = 20 dB

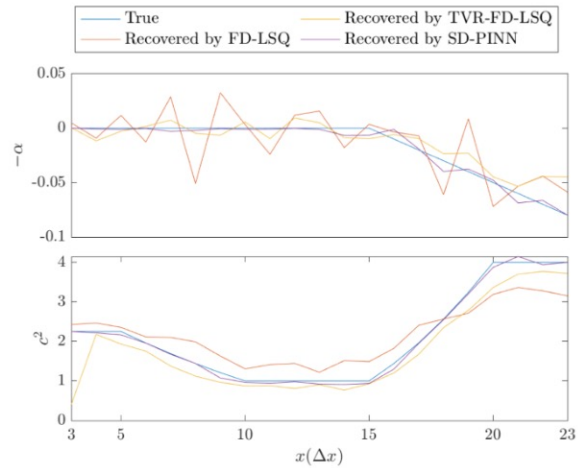




$$U_{tt} = -\alpha_m U_t + c_m^2 U_{xx}$$

SNR = 30 dB





		SD-PINN	FD-LSQ	TVR-FD-LSQ
Wave Eq. w/o attenuation	c^2	1.25×10^{-2}	7.87×10^{-1}	1.49
Wave Eq. w. attenuation	$-\alpha$	1.80×10^{-5}	5.31×10^{-4}	1.84×10^{-4}
	c^2	5.41×10^{-3}	2.30×10^{-1}	2.52×10^{-1}

The MSEs between the true PDE coefficients and the recovered ones from various methods.

Two baselines:

FD-LSQ: First compute the PDE terms by Finite Difference, then fix the term U_{tt} as the left-hand side, and finally compute the PDE coefficients for the right-hand side terms (U_{xx} and U_t) by least squares regression.

TVR-FD-LSQ: First compute the PDE terms by Total-Variation Regularized Finite Difference, then fix the term U_{tt} as the left-hand side, and finally compute the PDE coefficients for the right-hand side terms (U_{xx} and U_t) by least squares regression.

Summary:

1, The proposed SD-PINN can recover spatially-dependent PDE coefficients.

2, The sign information of the PDE coefficients is exploited to help training the SD-PINN.

3, The SD-PINN is robust to noise. Because it uses the neural network parameters to compute the PDE terms via automatic differentiation, and the neural network parameters are updated not only according to the noisy measurements, but also constrained by physical laws.

4, The noise-robustness of the SD-PINN is verified by multiple numerical experiments.

Key reference:

M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys., vol. 378, pp. 686–707, 2019.

Thank You!