Learned iterative solvers for the Helmholtz equation

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

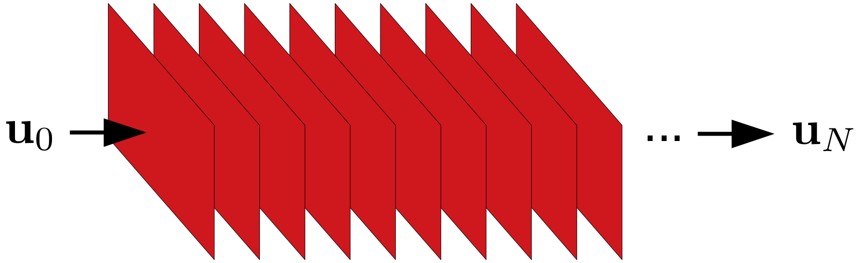
Georgia Institute of Technology

SLIM

EAGE
London, 06/06/2019

Neural nets and iterative schemes in computational maths

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

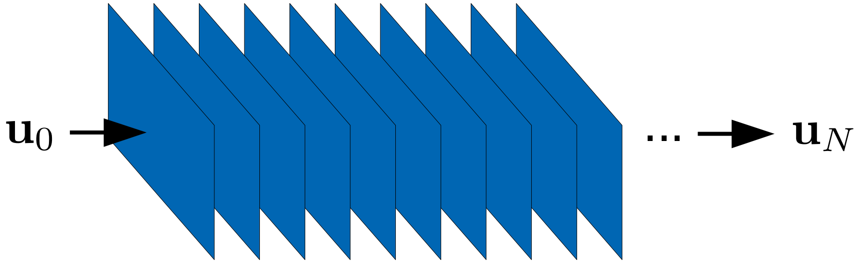


$$\partial_t \mathbf{u}(t) = F(t, \mathbf{u}(t)) \qquad \mathbf{u}(t + \Delta t) \approx \mathbf{u}(t) + a(W(t) * \mathbf{u}(t) + b(t))$$

[Haber and Ruthotto, 2017]

Neural nets and iterative schemes in computational maths

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



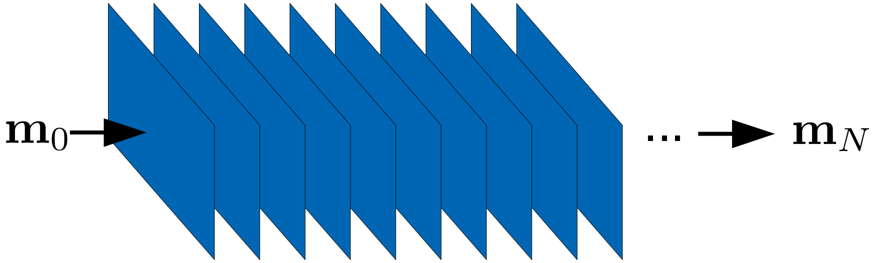
$$\mathbf{m}\,\partial_{tt}\mathbf{u} - \Delta\mathbf{u} = \mathbf{f} \qquad \mathbf{u}(t+\Delta t) \approx 2\mathbf{u}(t) - \mathbf{u}(t-\Delta t) + \Delta t^2/\mathbf{m}\,(\Delta\mathbf{u}(t) + \mathbf{f}(t))$$

...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**...



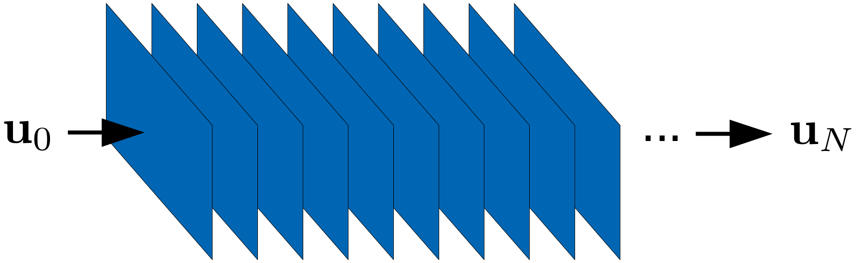
$\min_{\mathbf{m}} J(\mathbf{m}) \qquad \mathbf{m}_{i+1} = \mathbf{m}_i - \alpha \nabla_{\mathbf{m}_i} J$

...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**...

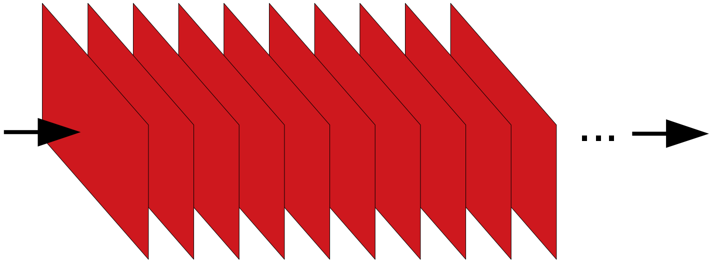


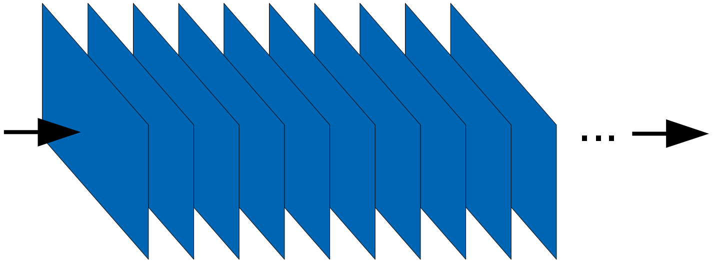
$$A\mathbf{u} = \mathbf{b} \qquad \mathbf{u}_{i+1} = \mathbf{u}_i + \sum_{j \leq i} \alpha_j \mathbf{r}_j$$

...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

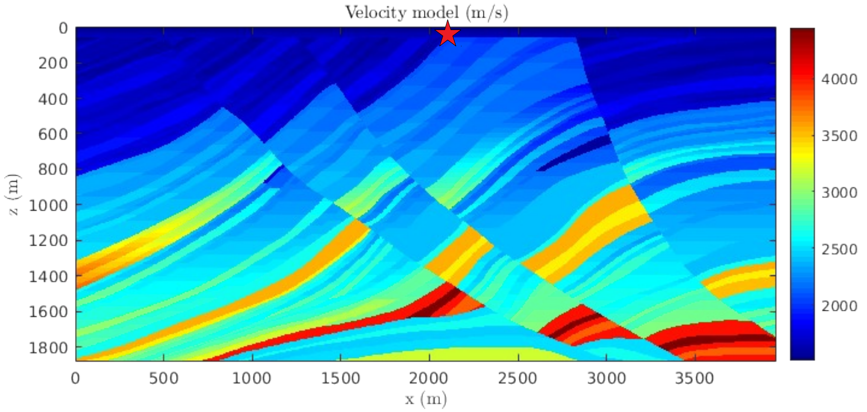
This work:
(neural net)-augmented physics/maths

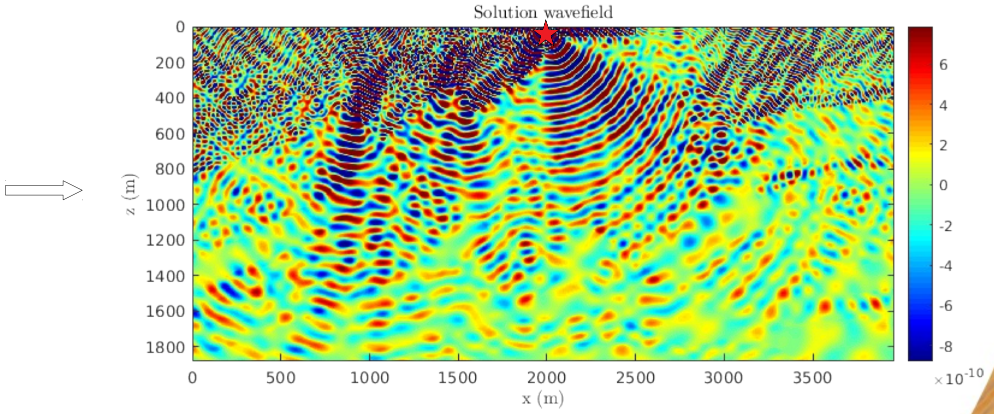
Helmholtz equation

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

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

$$H[\mathbf{m}] = -\omega^2 \mathbf{m} - \Delta, \qquad H[\mathbf{m}]\mathbf{u} = \mathbf{f}$$





Helmholtz equation: classical solution methods

Classical solution methods:

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

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

Complexity	2D	3D
# grid points	n^2	n^3
factorization	$n_f n^3$	$n_f n^6$
application	$n_s n_f n^2 \log n$	$n_s n_f n^4 \log n$

Helmholtz equation: classical solution methods

Classical solution methods:

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

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \sum_{j \leq i} \alpha_j \mathbf{r}_j, \quad \mathbf{r}_j = \mathbf{f} - H[\mathbf{m}]\,\mathbf{u}_j$$

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]):

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \sum_{j \leq i} \alpha_j \mathbf{r}_j, \quad \mathbf{r}_j = \mathbf{f} - H[\mathbf{m}]\,\mathbf{u}_j$$

Need **pre-conditioning**!

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

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

...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...

Poisson equation	[Tang et al., 2017], [Tompson et al., 2017], [Farimani et al. 2017], [Zhang et al., 2018], [Hsieh et al., 2019]
Laplace equation	[Sharma et al., 2017]
Schrodinger equation	[Mills et al., 2017]
Fluid dynamics	[Guo et al., 2016], [Yang et al., 2016], [Chu and Thuerey, 2017], [Kutz, 2017], [Singh et al., 2017]
Black-Scholes	[Sirignano and Spiliopoulos, 2018]

Helmholtz equation: Krylov net training setup

Goal:

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

$$F : \mathcal{M} \to \mathcal{U},\; F(\mathbf{m}) = (H[\mathbf{m}])^{-1}\,\mathbf{f} \quad \Longleftrightarrow \quad F_\theta : \mathcal{M} \to \mathcal{U},\; F_\theta(\mathbf{m}) \approx (H[\mathbf{m}])^{-1}\,\mathbf{f}$$

Candidate loss functions:

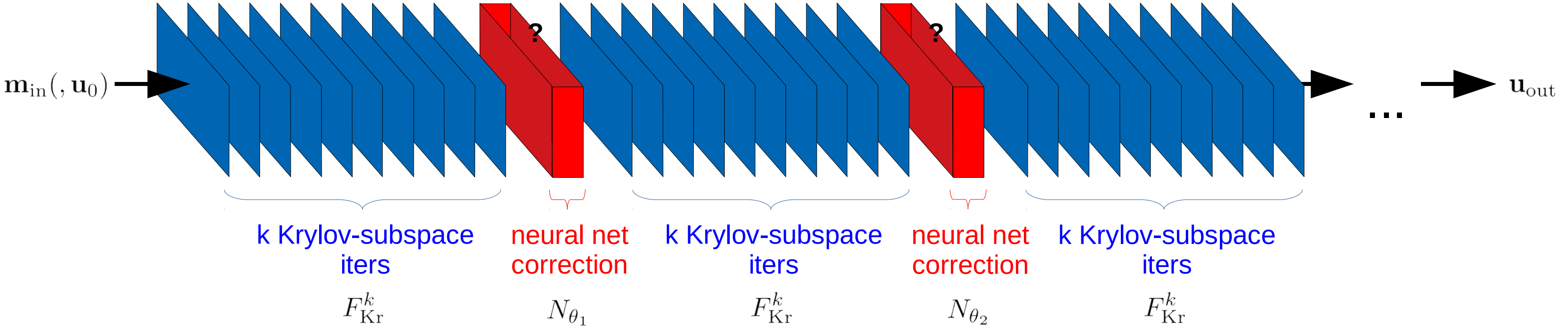
$$F_{\theta^*} = F_{\arg\min_\theta \; L(\theta)},$$

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

- Training with stochastic gradient descent algorithms (ADAM, [Kingma and Ba, 2015])

Helmholtz equation: Krylov net structure

Main idea: **intersperse** Krylov-subspace “nets” and neural nets...



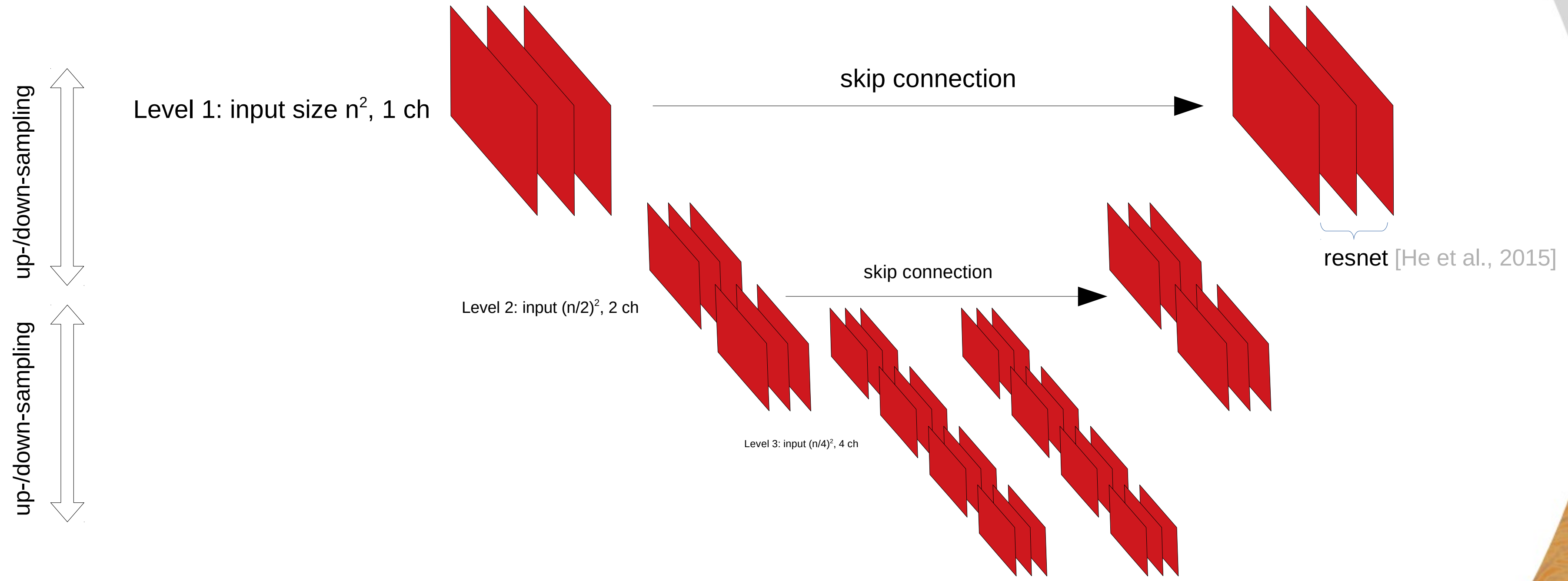
$$F_{(\theta_1,\ldots,\theta_N)} : \mathcal{M} \to \mathcal{U}, \qquad F_{(\theta_1,\ldots,\theta_N)}(\mathbf{m}) = \begin{cases} F^k_{\text{Kr}}(\mathbf{m}, N_{\theta_N} \circ F_{(\theta_1,\ldots,\theta_{N-1})}(\mathbf{m})), & N \geq 1 \\ F^k_{\text{Kr}}(\mathbf{m}, \underbrace{\mathbf{u}_0}_{\text{fixed}}), & N = 0 \end{cases}$$

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

$$N_{\theta_i} : \mathcal{U} \to \mathcal{U}$$

Helmholtz equation: Krylov net structure

2-D net correction architecture (“Unet”) ~ multigrid
(e.g. [Ke et al., 2017], [He and Xu, 2019]):



Complexity ~ $O(n^2 \log n)$

Helmholtz equation: Krylov net structure

2-D net correction architecture, **two-grid sketch**:

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

activation function (e.g., ReLU)

resnet

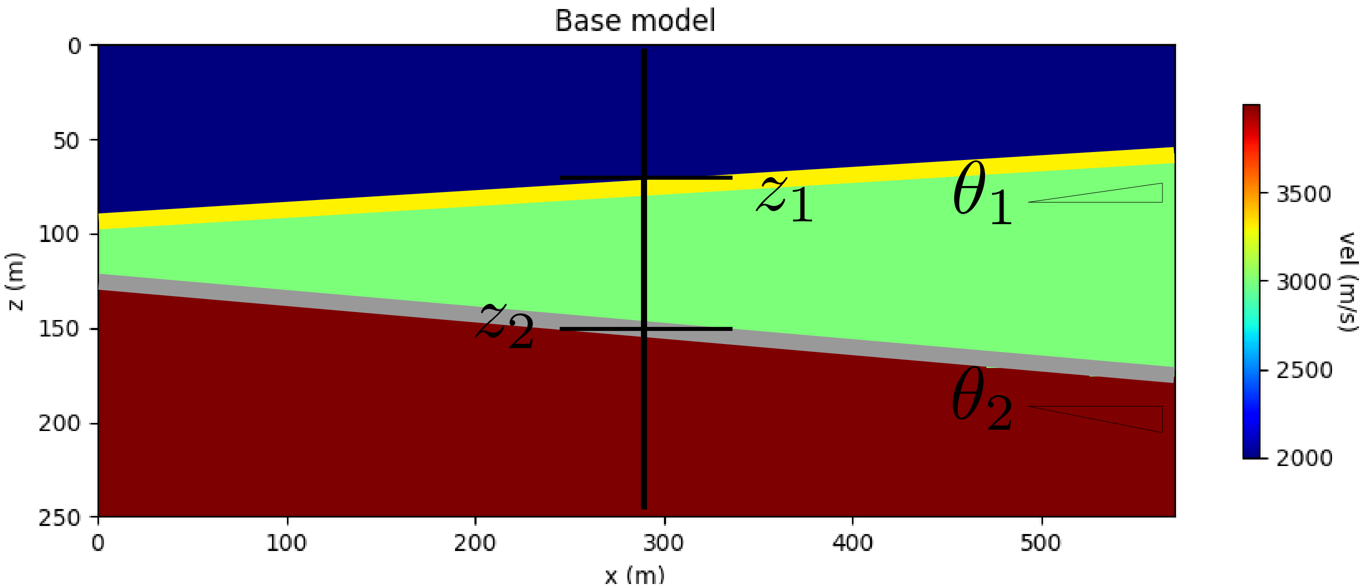
Restriction (many channels!): $\mathbf{x}_{\mathrm{ch}_i}^{2h} \leftarrow R_h^{2h}(W_{\mathrm{ch}_i}^h * \mathbf{x}^h + b_{\mathrm{ch}_i}^h), \quad \text{for } i = 1, \ldots, N_{\mathrm{ch}}$

Smoothing (coarse grid): $\mathbf{x}_{\mathrm{ch}_i}^{2h} \leftarrow \mathbf{x}_{\mathrm{ch}_i}^{2h} + a(\sum_j W_{k,\mathrm{ch}_i,\mathrm{ch}_j}^{2h} * \mathbf{x}_{\mathrm{ch}_j}^{2h} + b_{k,\mathrm{ch}_i}^{2h}), \quad \text{for } i, k, \ldots$

Prolongation: $\mathbf{x}^h \leftarrow \mathbf{x}^h + P_{2h}^h \sum_i (W_{\mathrm{ch}_i}^{2h} * \mathbf{x}_{\mathrm{ch}_i}^{2h} + b_{\mathrm{ch}_i}^{2h})$

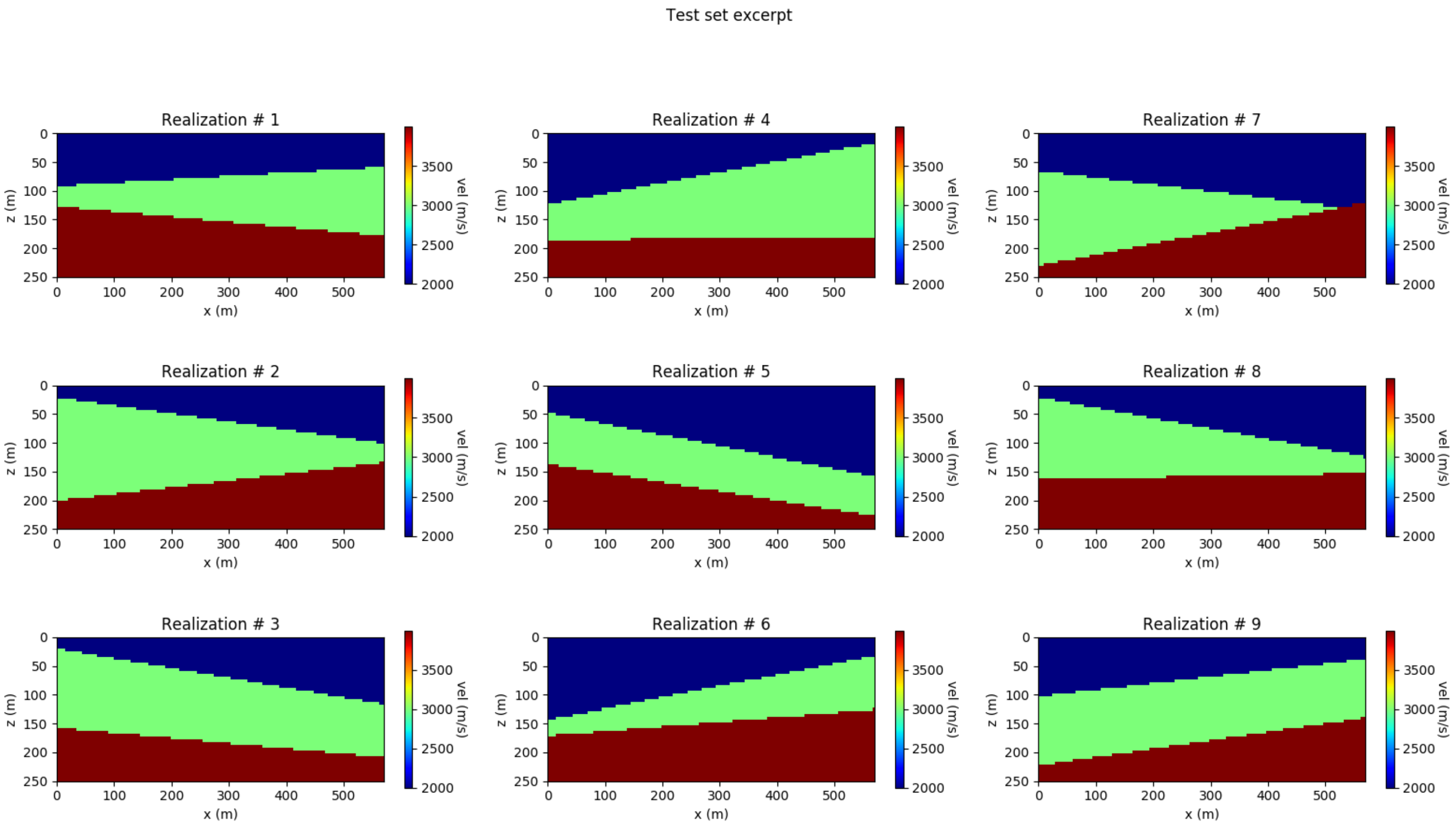
Post-smoothing (fine grid): $\mathbf{x}^h \leftarrow \mathbf{x}^h + a(W_l^h * \mathbf{x}^h + b_l^h), \quad \text{for } l = 1, \ldots, N$

Example 1: layered model distribution (fixed source and frequency)

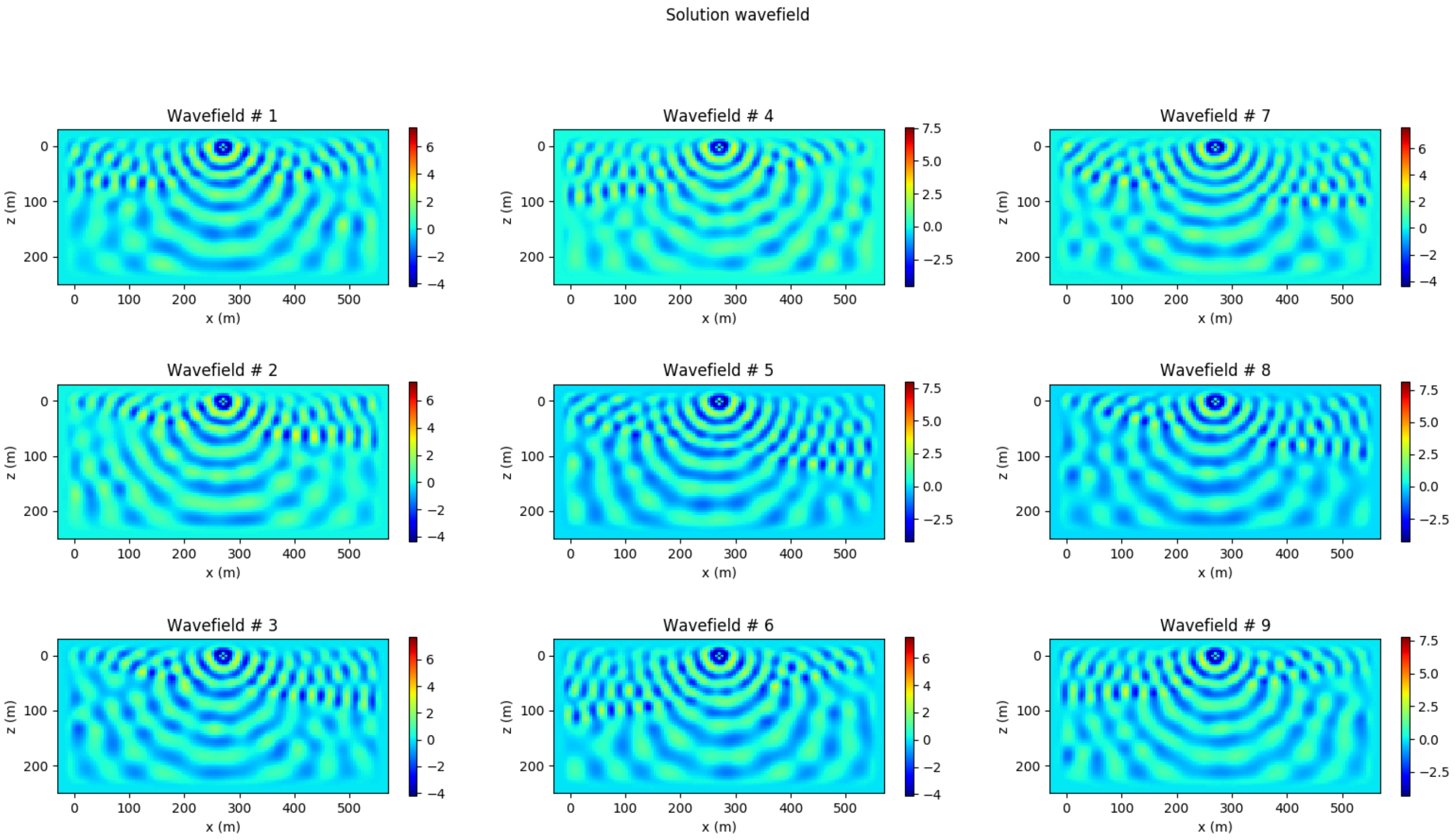


$$z \sim z^0 + U(-a, a) \qquad \theta \sim U(-b, b)$$

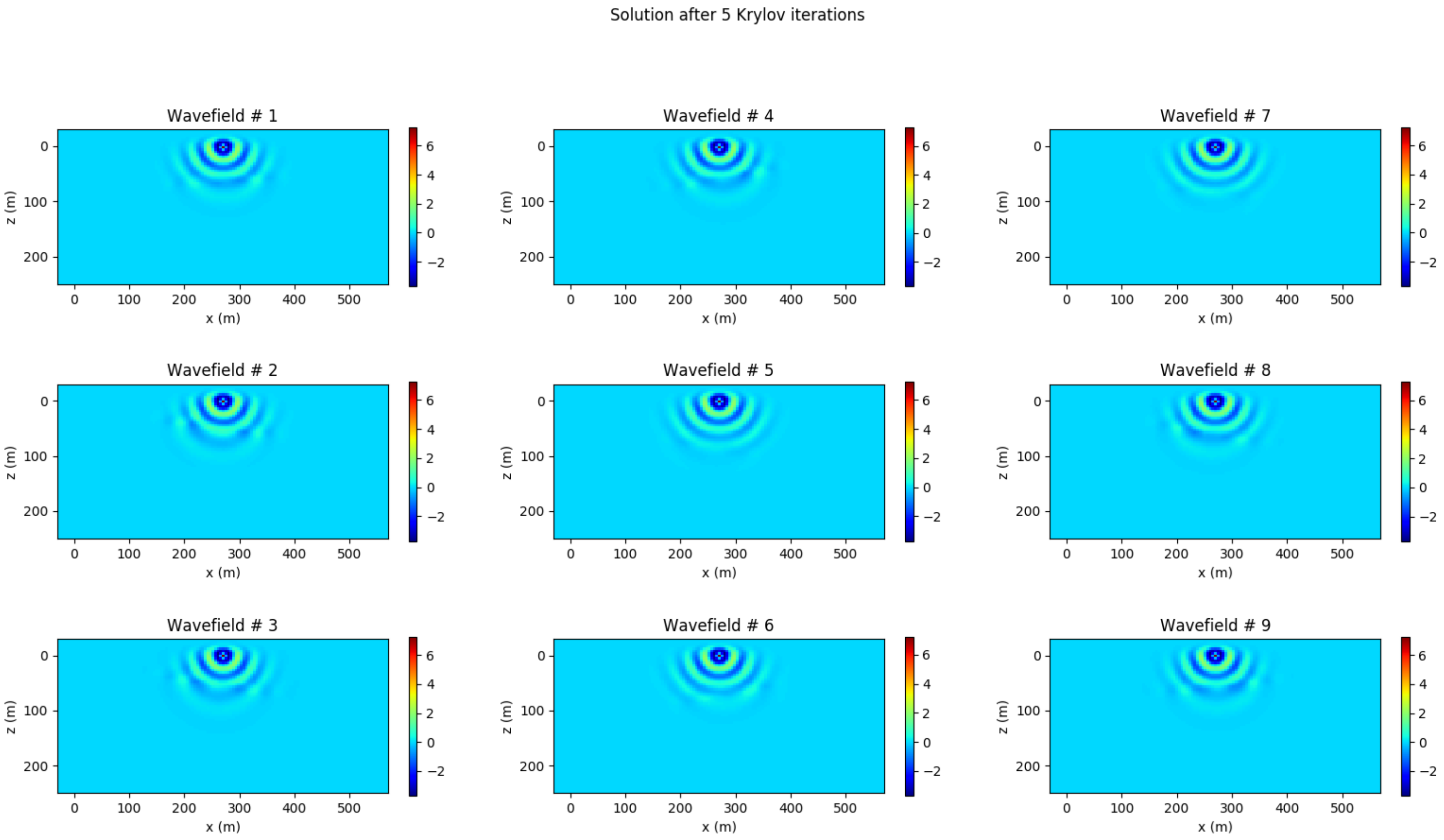
Example 1: layered model distribution (train size: 1024, test size: 16)



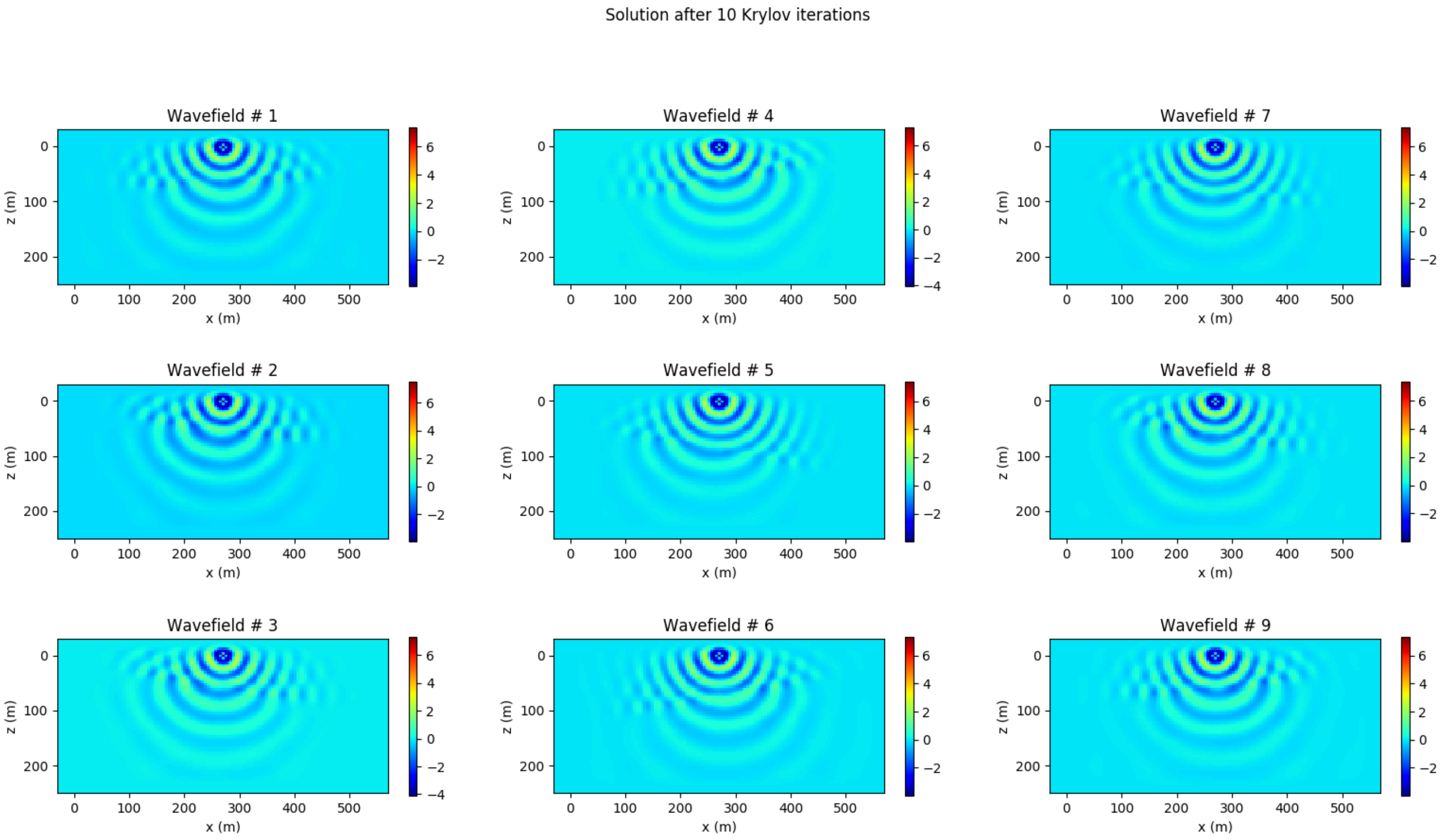
Example 1: solution distribution at 60 Hz (train size: 1024, test size: 16)



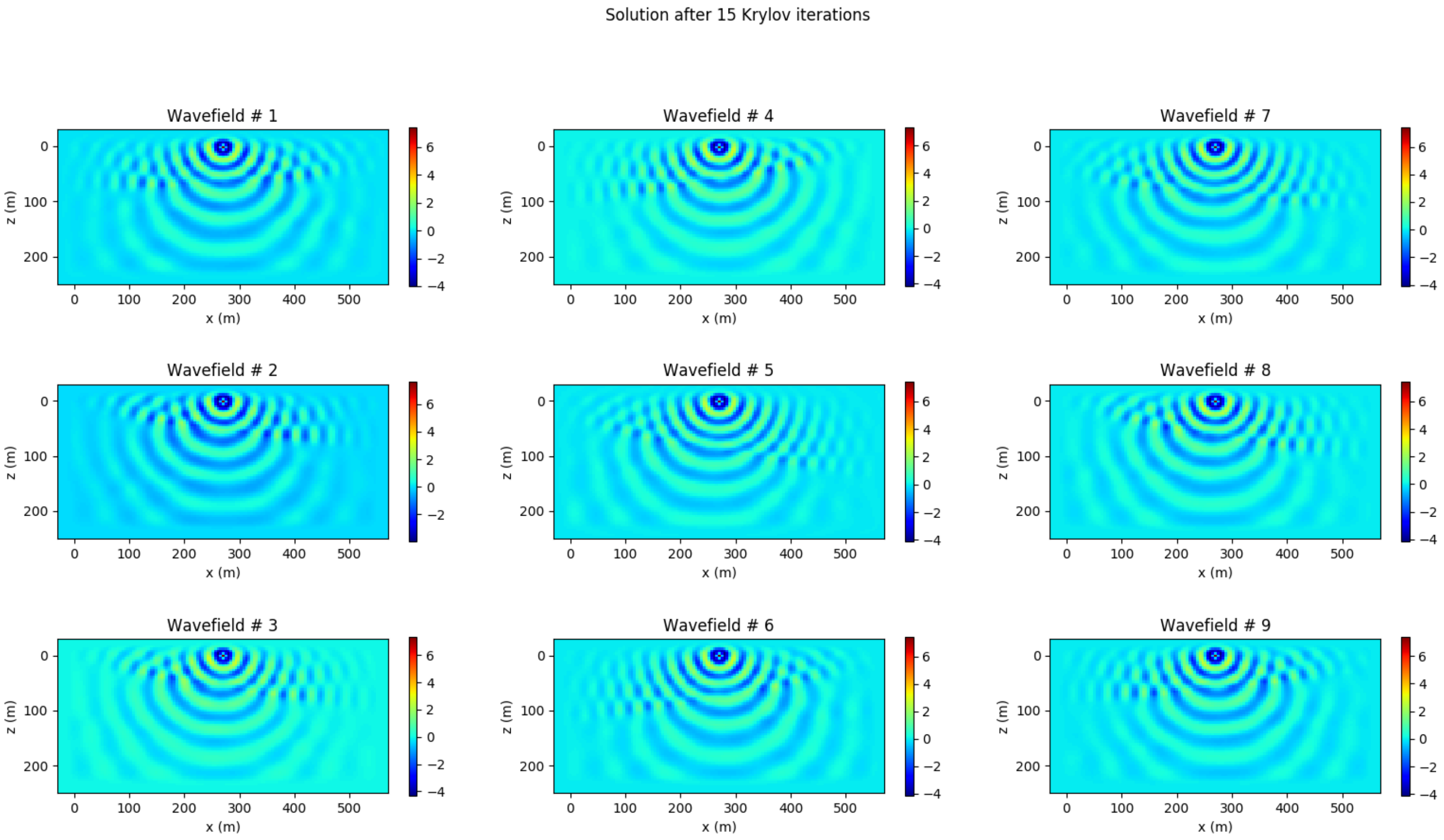
Example 1: approximated wavefield at 60 Hz (after 5 Krylov iterations)



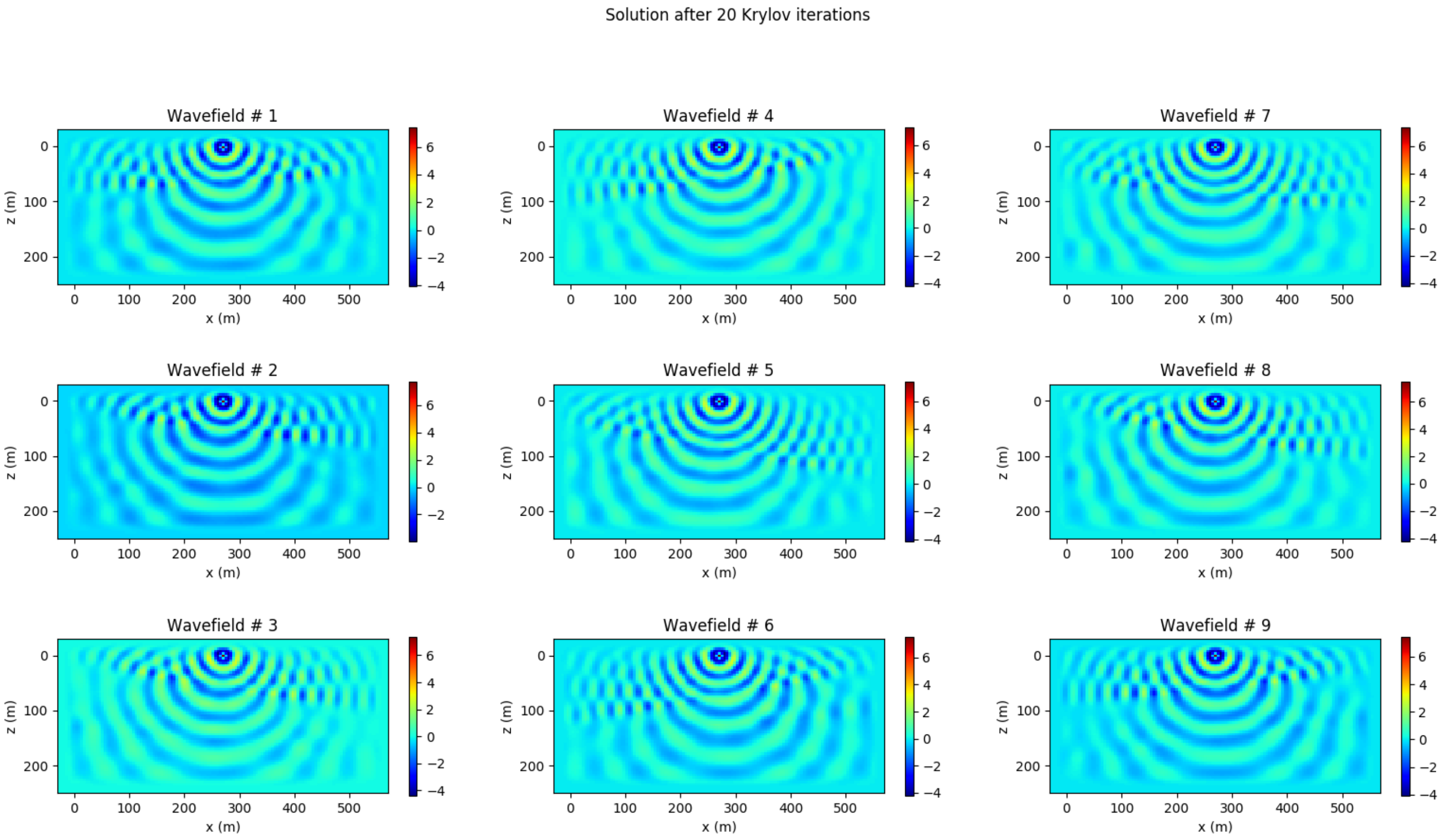
Example 1: approximated wavefield at 60 Hz (after 10 Krylov iterations)



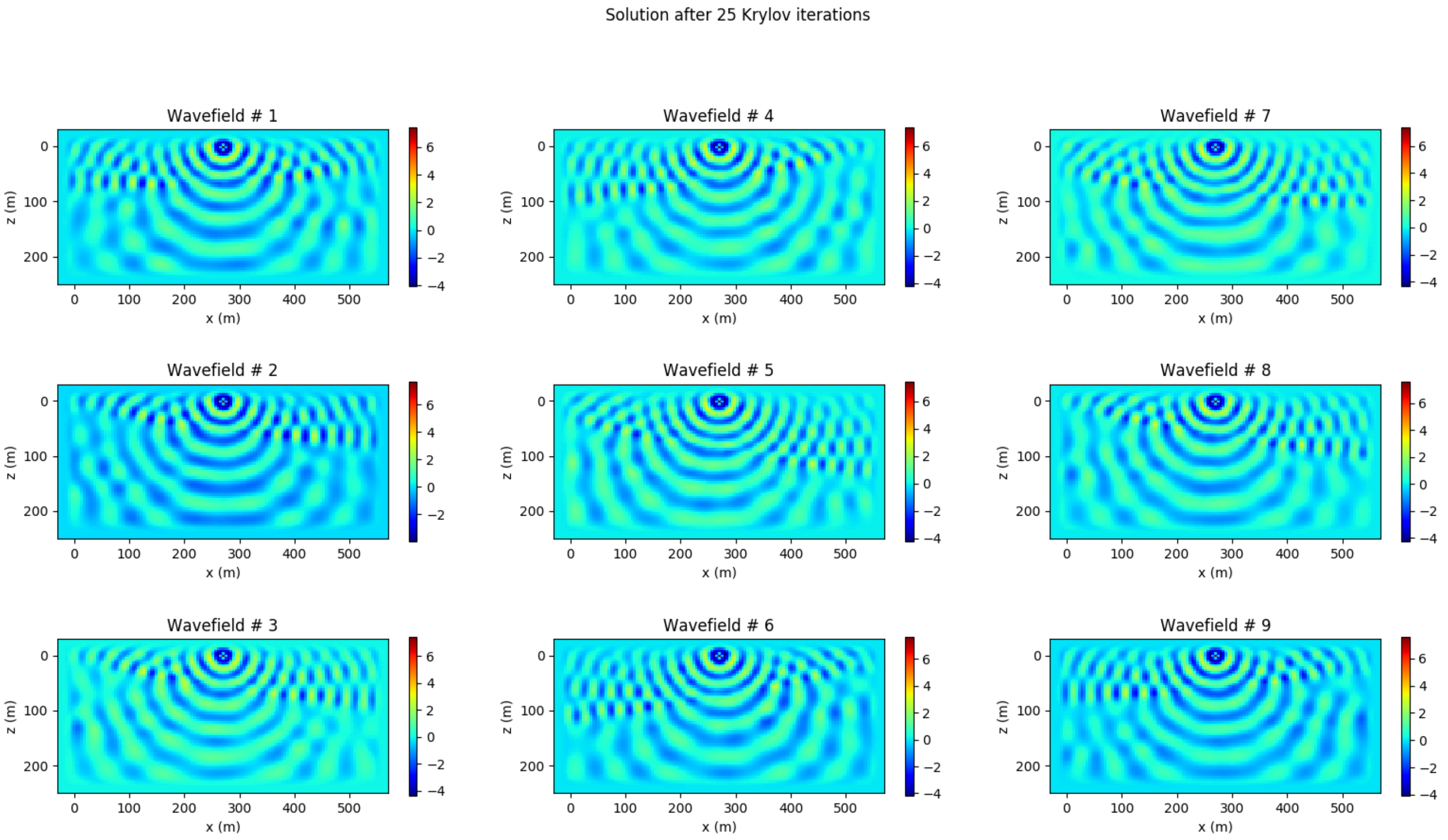
Example 1: approximated wavefield at 60 Hz (after 15 Krylov iterations)



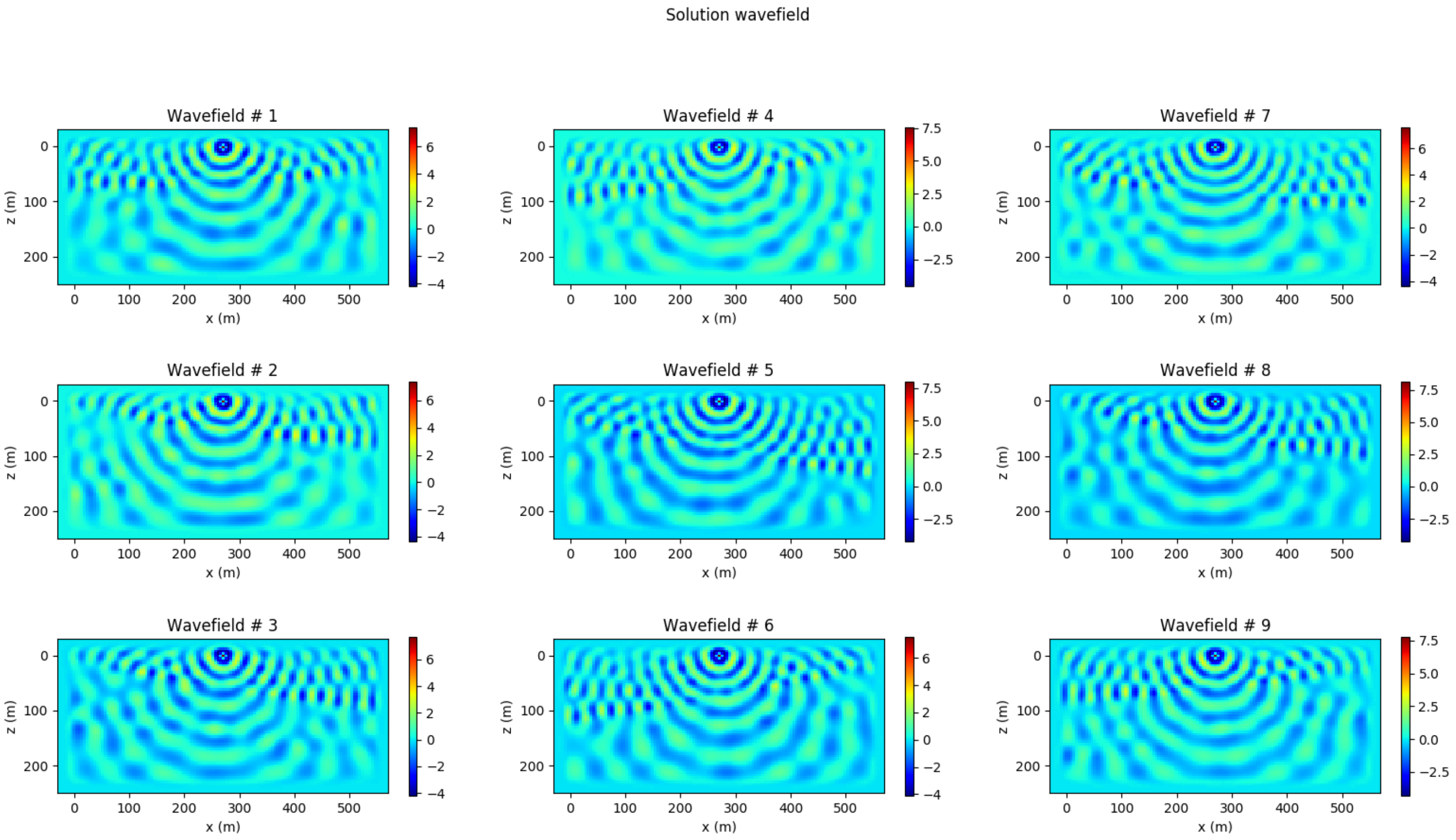
Example 1: approximated wavefield at 60 Hz (after 20 Krylov iterations)



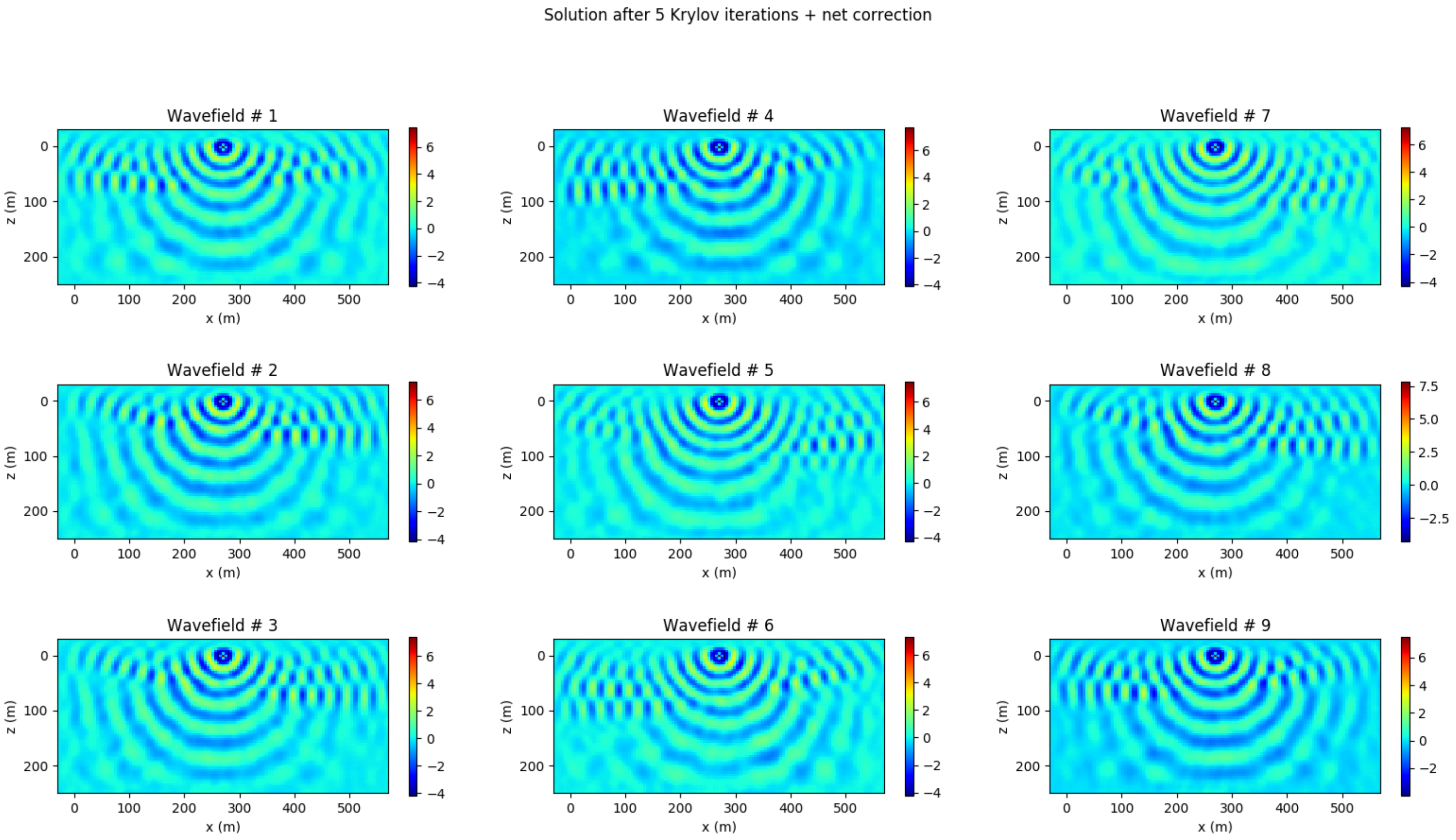
Example 1: approximated wavefield at 60 Hz (after 25 Krylov iterations)



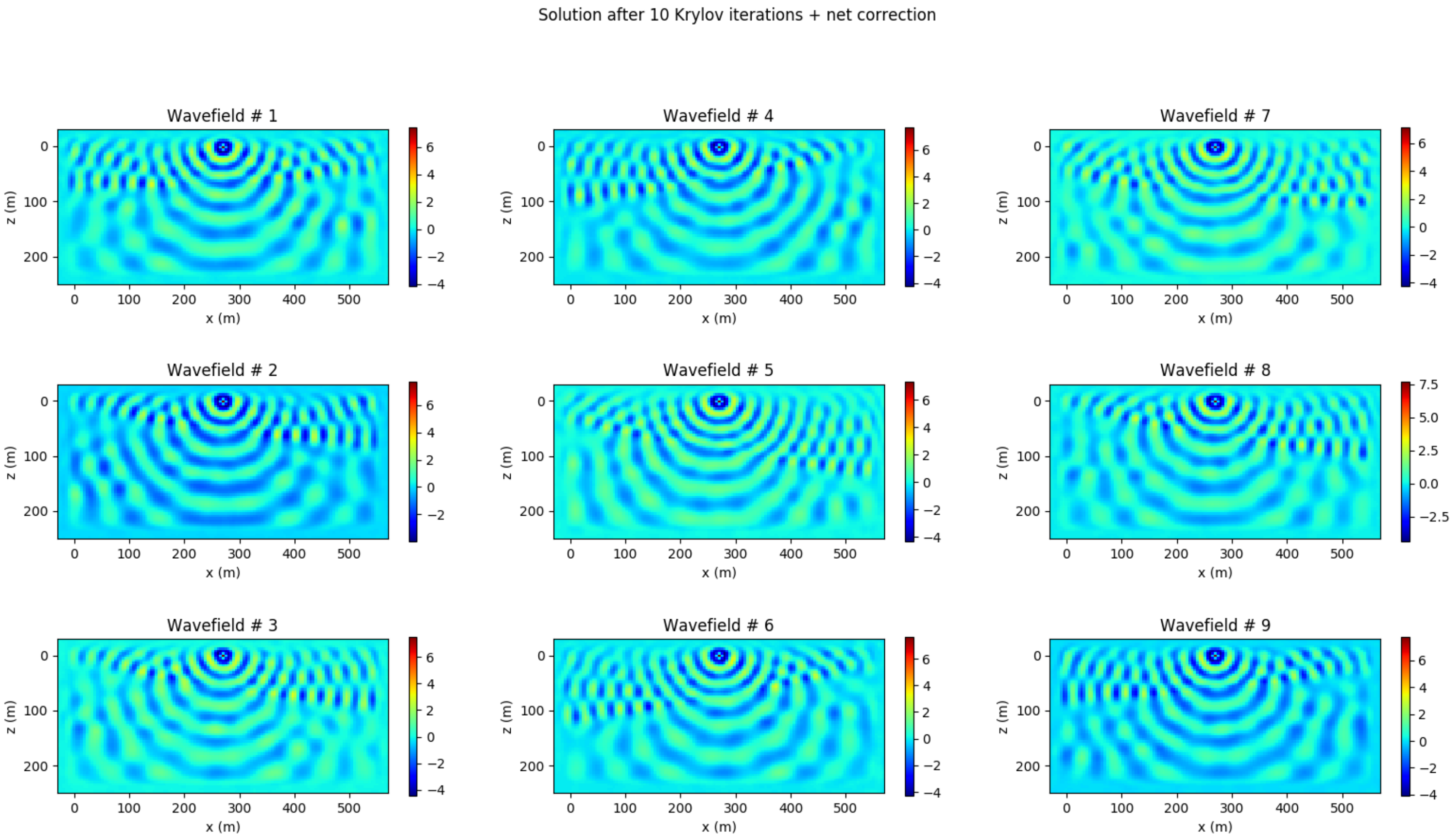
Example 1: solution distribution at 60 Hz (train size: 1024, test size: 16)



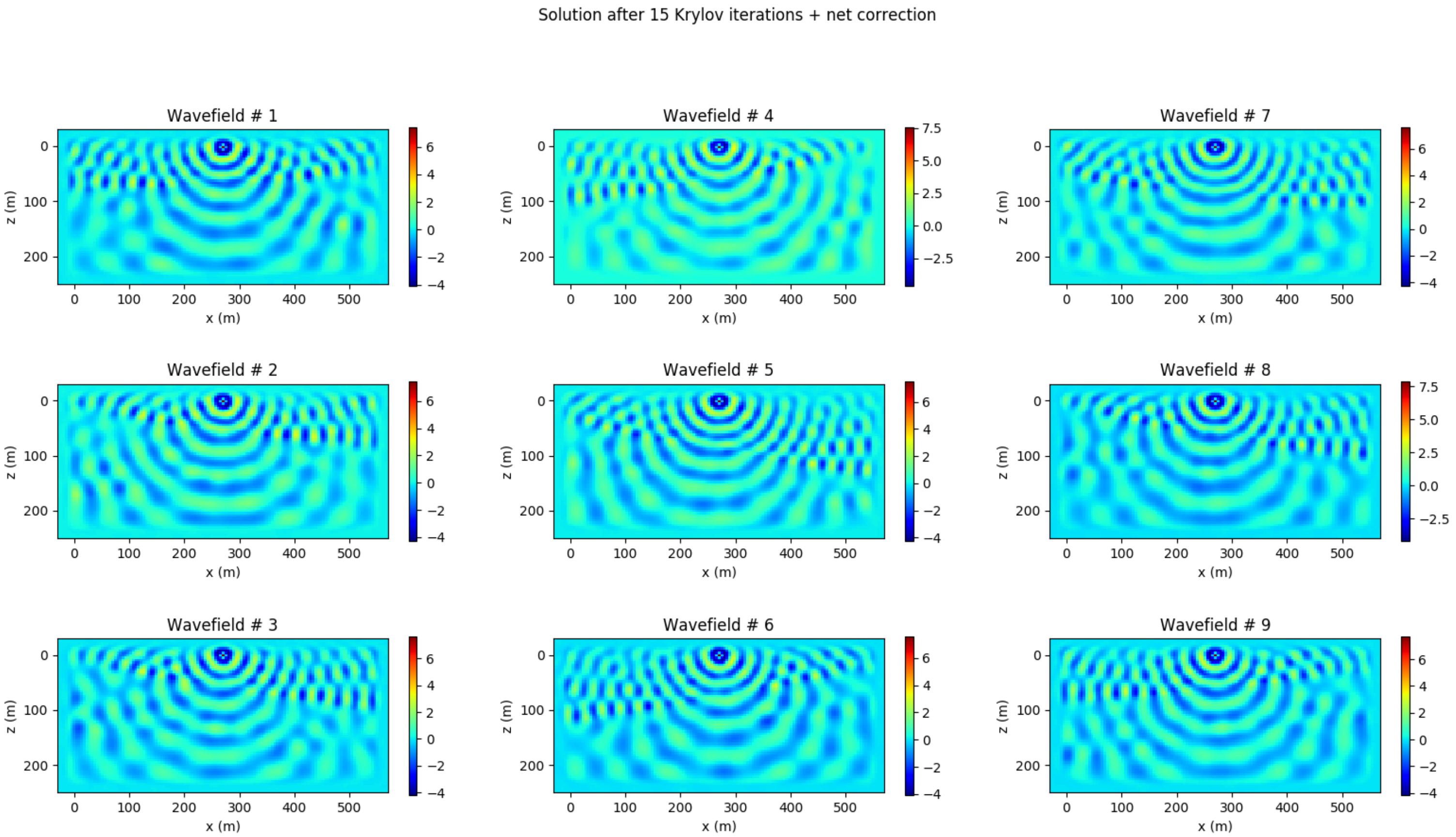
Example 1: approximated wavefield at 60 Hz (after 5 Krylov iterations + net)



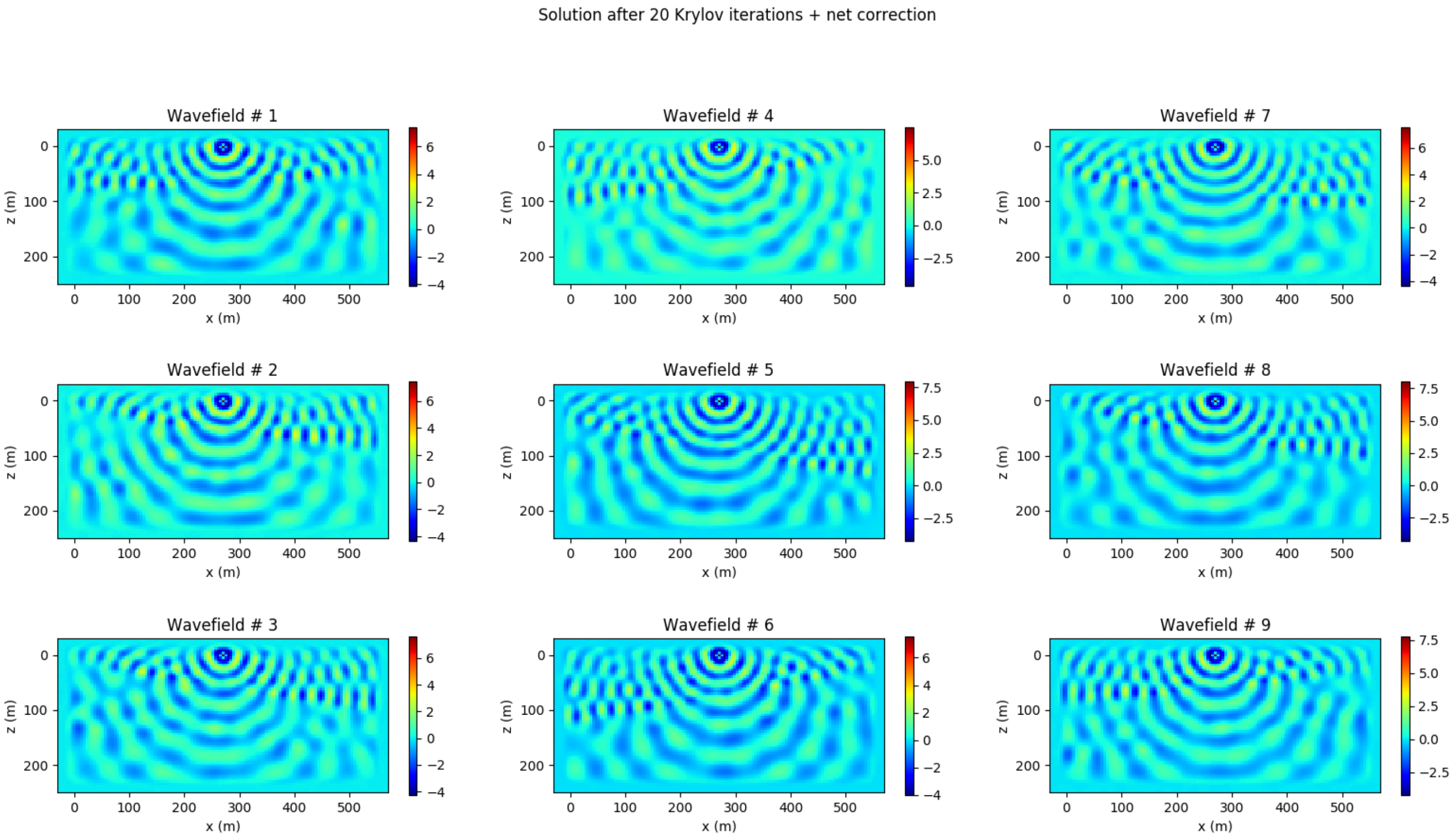
Example 1: approximated wavefield at 60 Hz (after 10 Krylov iterations + net)



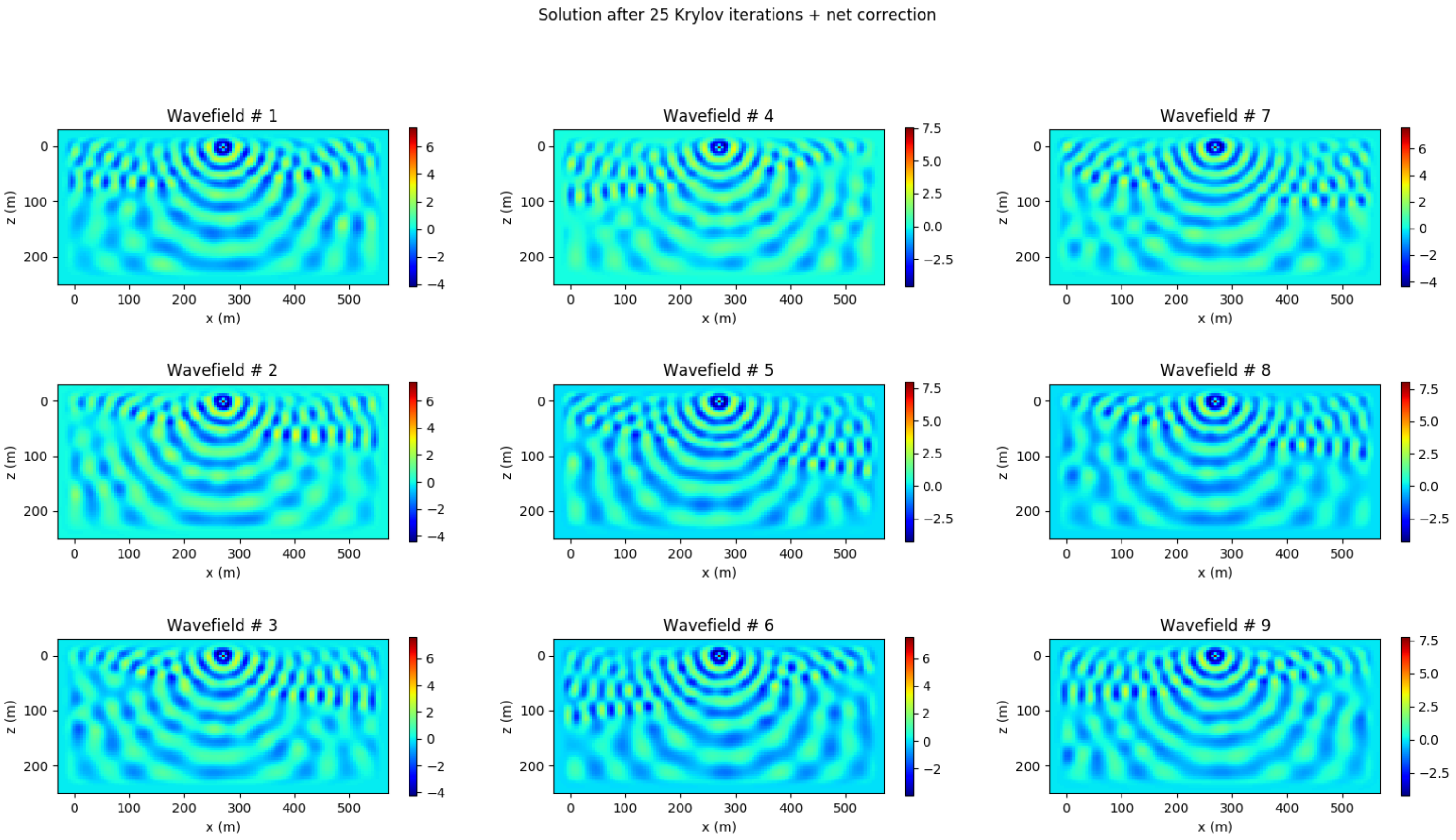
Example 1: approximated wavefield at 60 Hz (after 15 Krylov iterations + net)



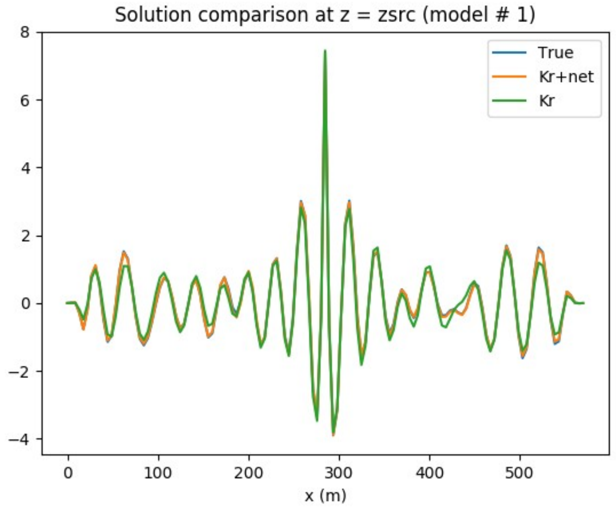
Example 1: approximated wavefield at 60 Hz (after 20 Krylov iterations + net)

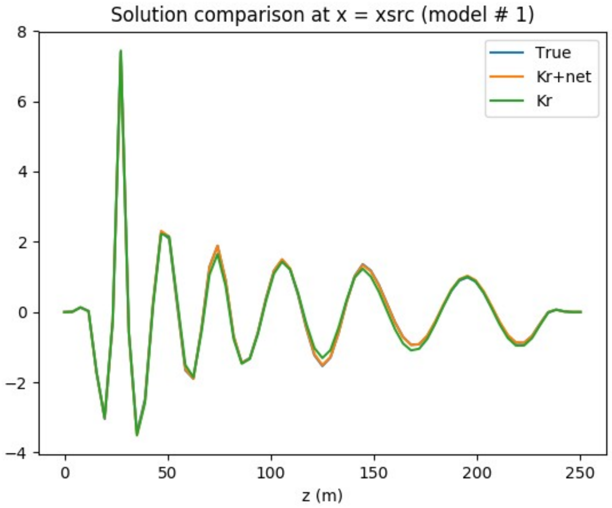


Example 1: approximated wavefield at 60 Hz (after 25 Krylov iterations **+ net**)

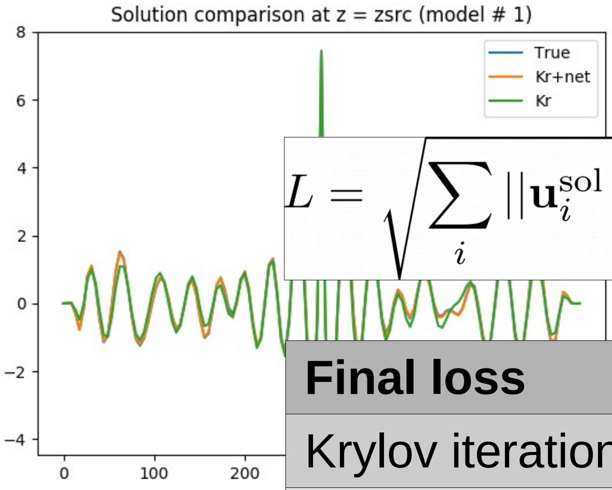


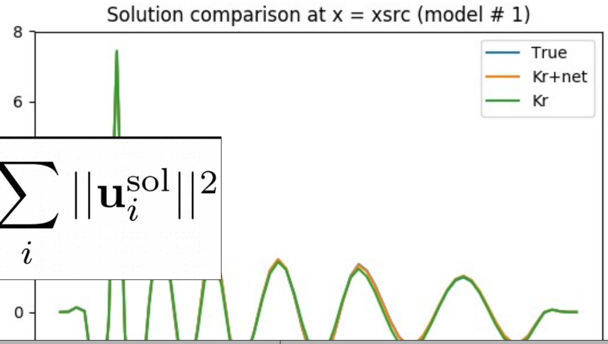
Example 1: solution trace comparison





Example 1: training/test errors

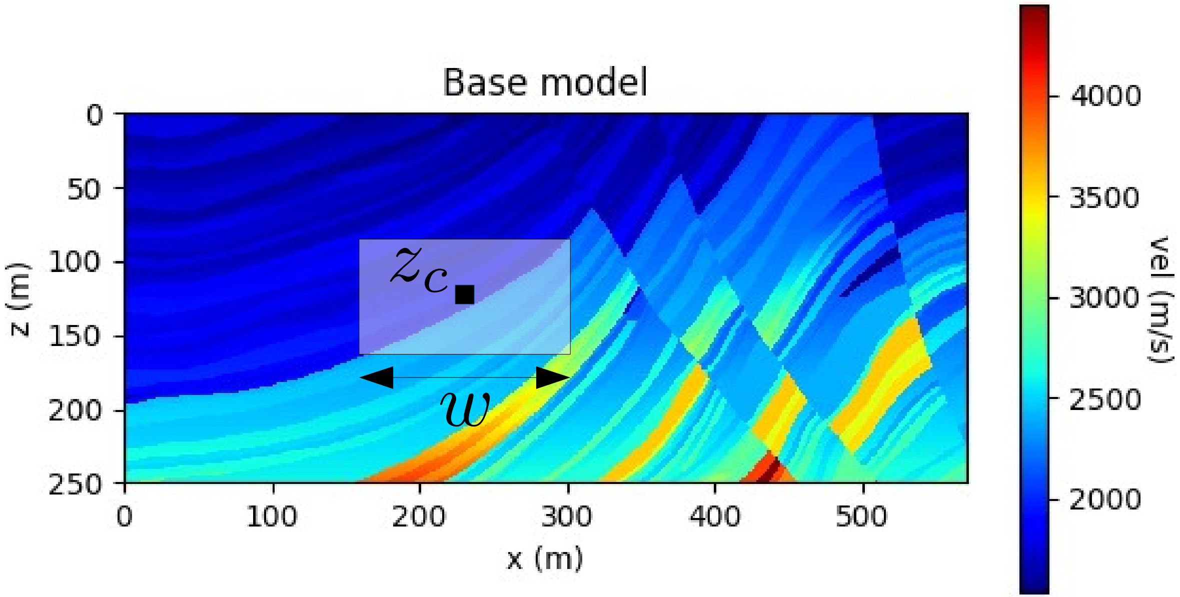




$$L = \sqrt{\sum_i ||\mathbf{u}_i^{\text{sol}} - \tilde{\mathbf{u}}_i||^2 / \sum_i ||\mathbf{u}_i^{\text{sol}}||^2}$$

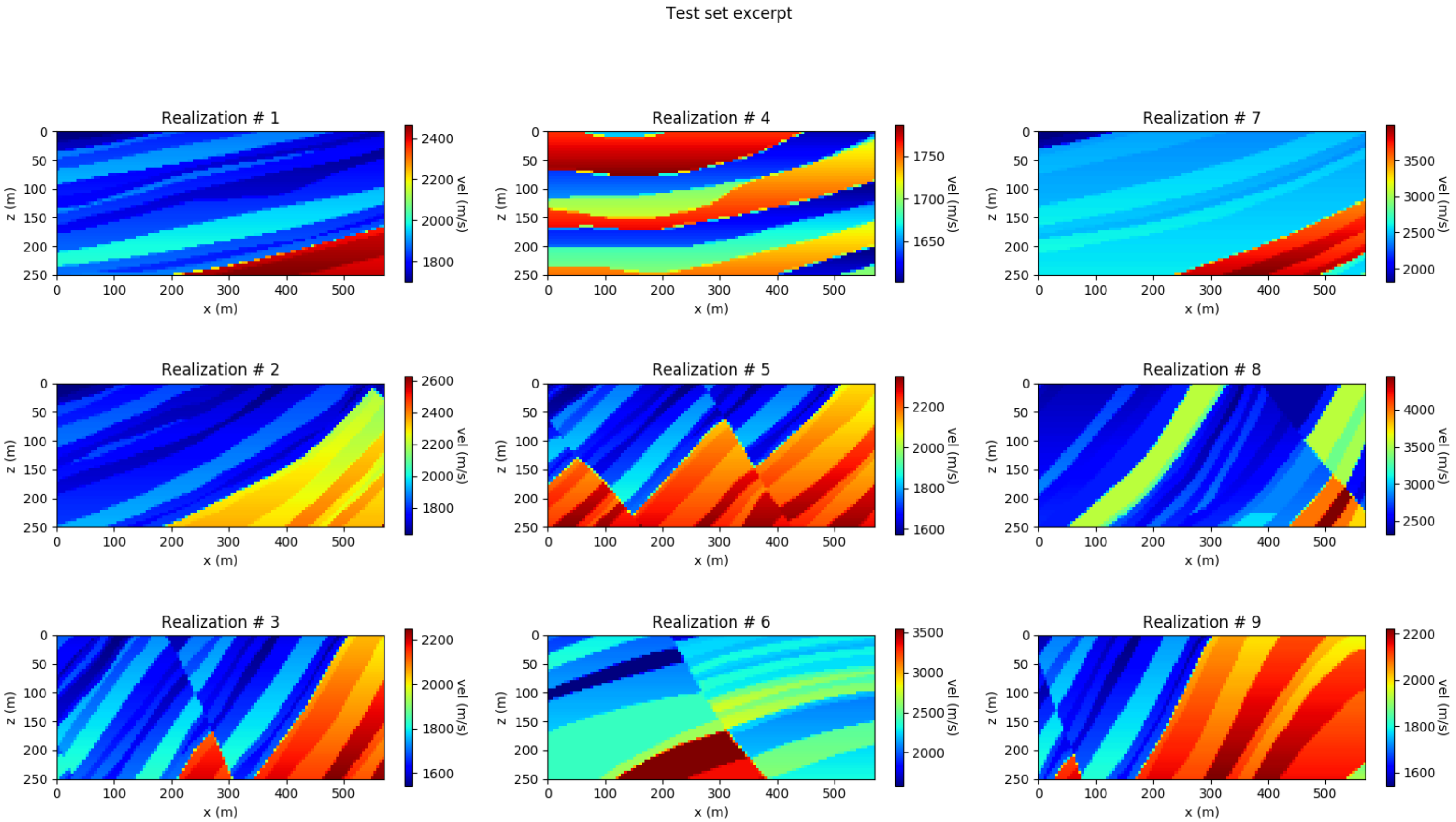
Final loss	Train error	Test error
Krylov iterations	23%	24%
Krylov net	3.9%	5.2%

Example 2: Marmousi-like distribution (fixed source and frequency)

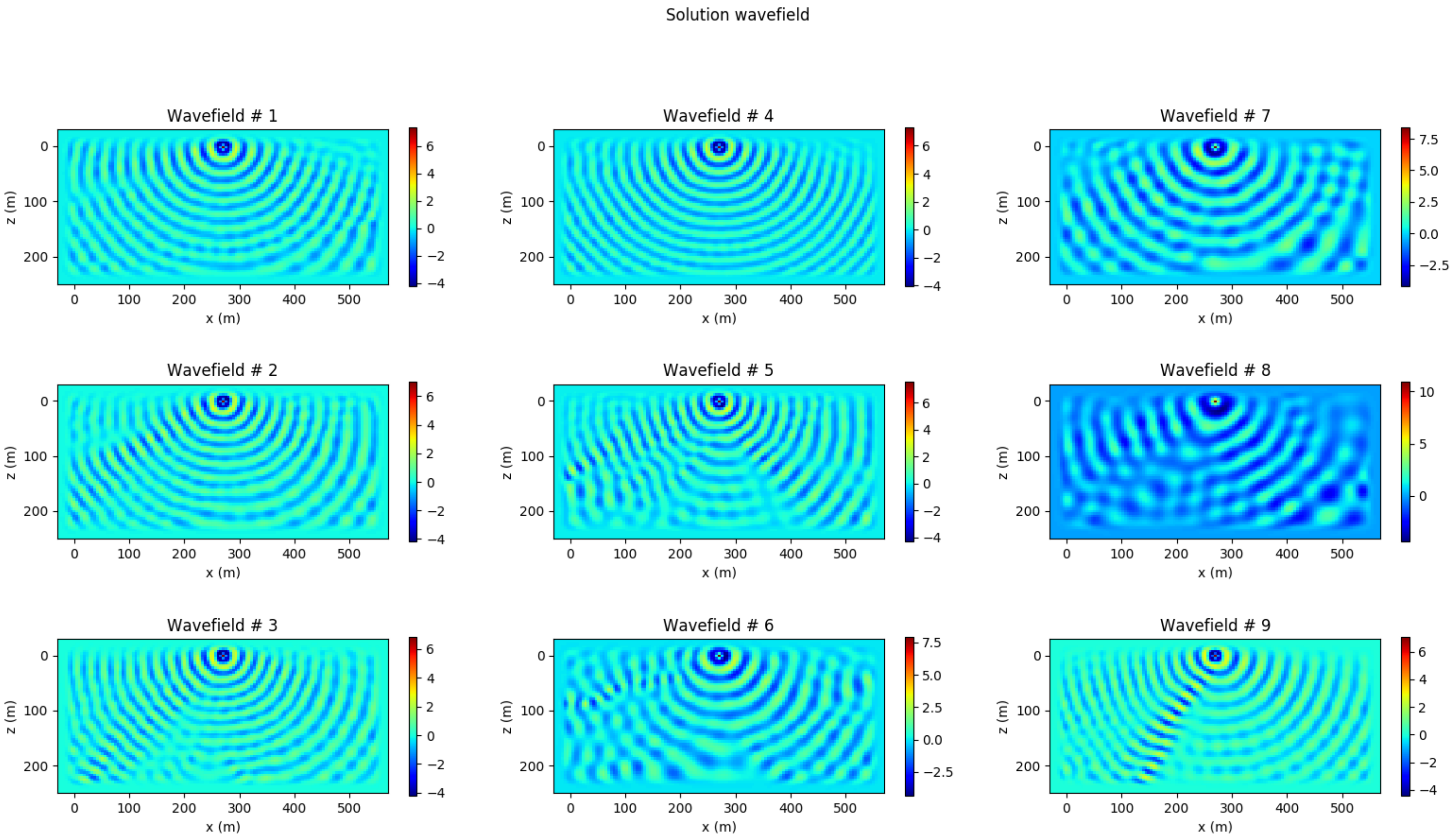


$z_c \sim U(z_0, z_{\text{end}})$ $\quad w \sim w_0 + U(-a, a)$

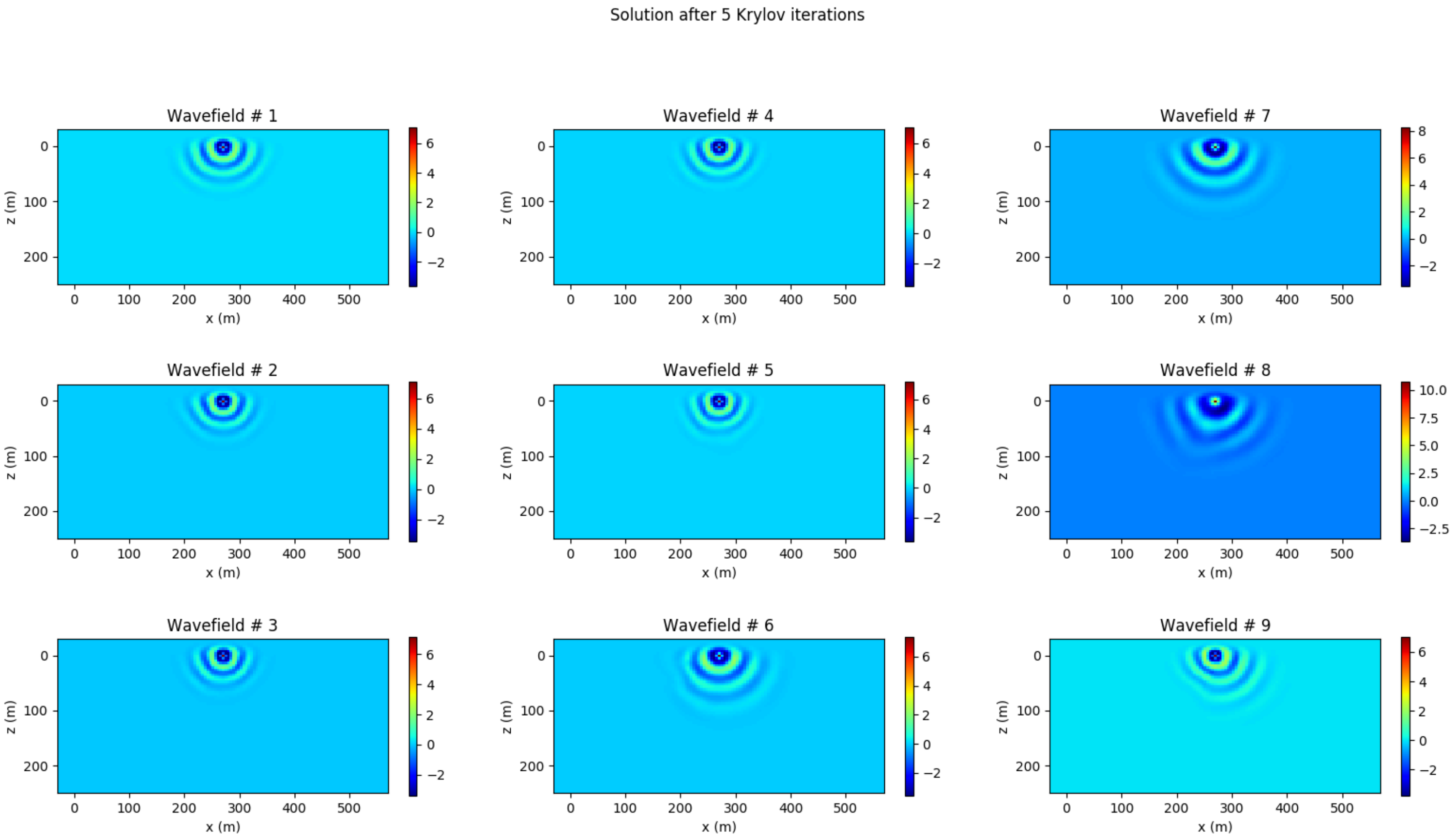
Example 2: Marmousi-like distribution (train size: 1024, test size: 16)



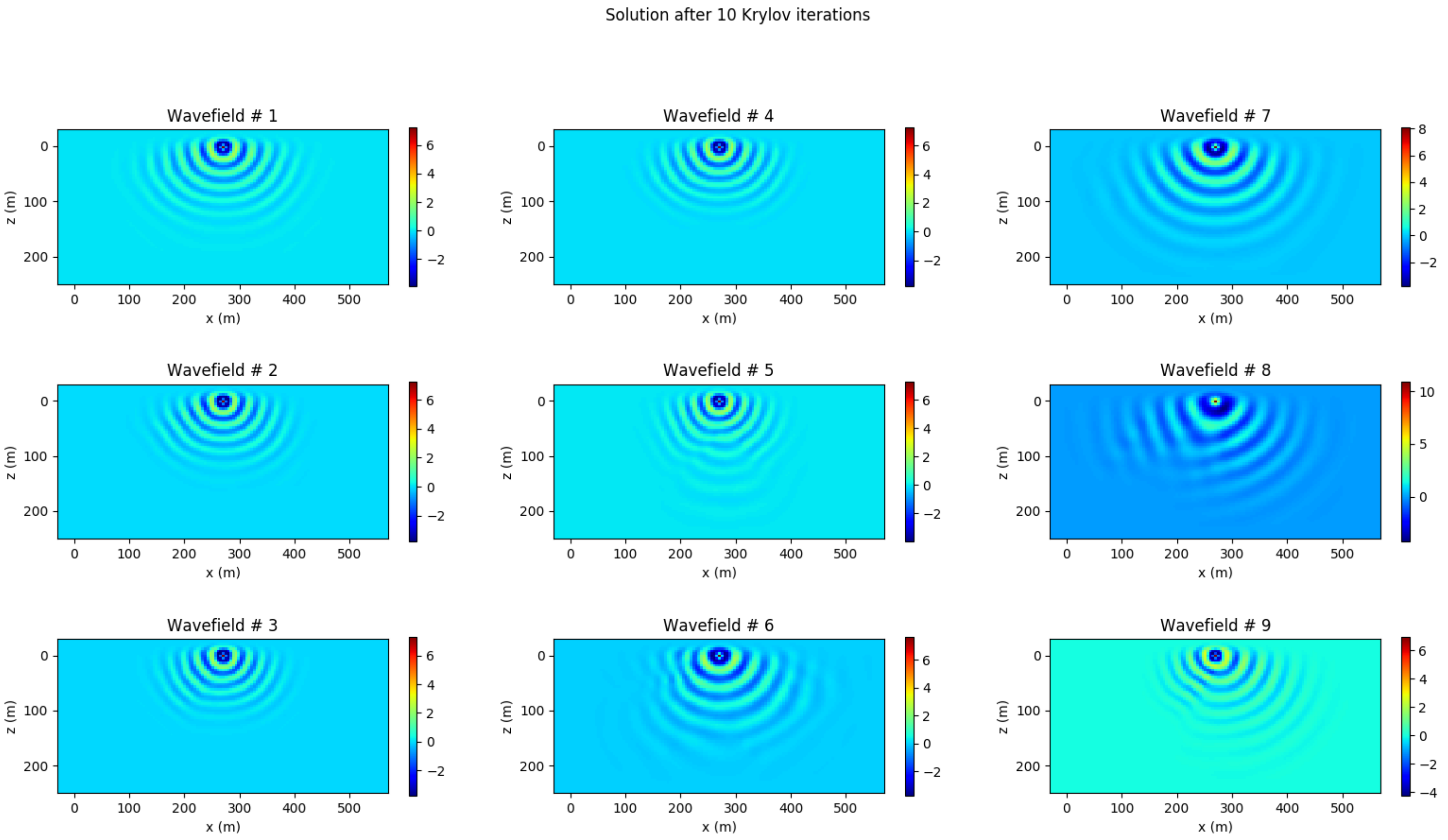
Example 2: solution distribution at 60 Hz (train size: 1024, test size: 16)



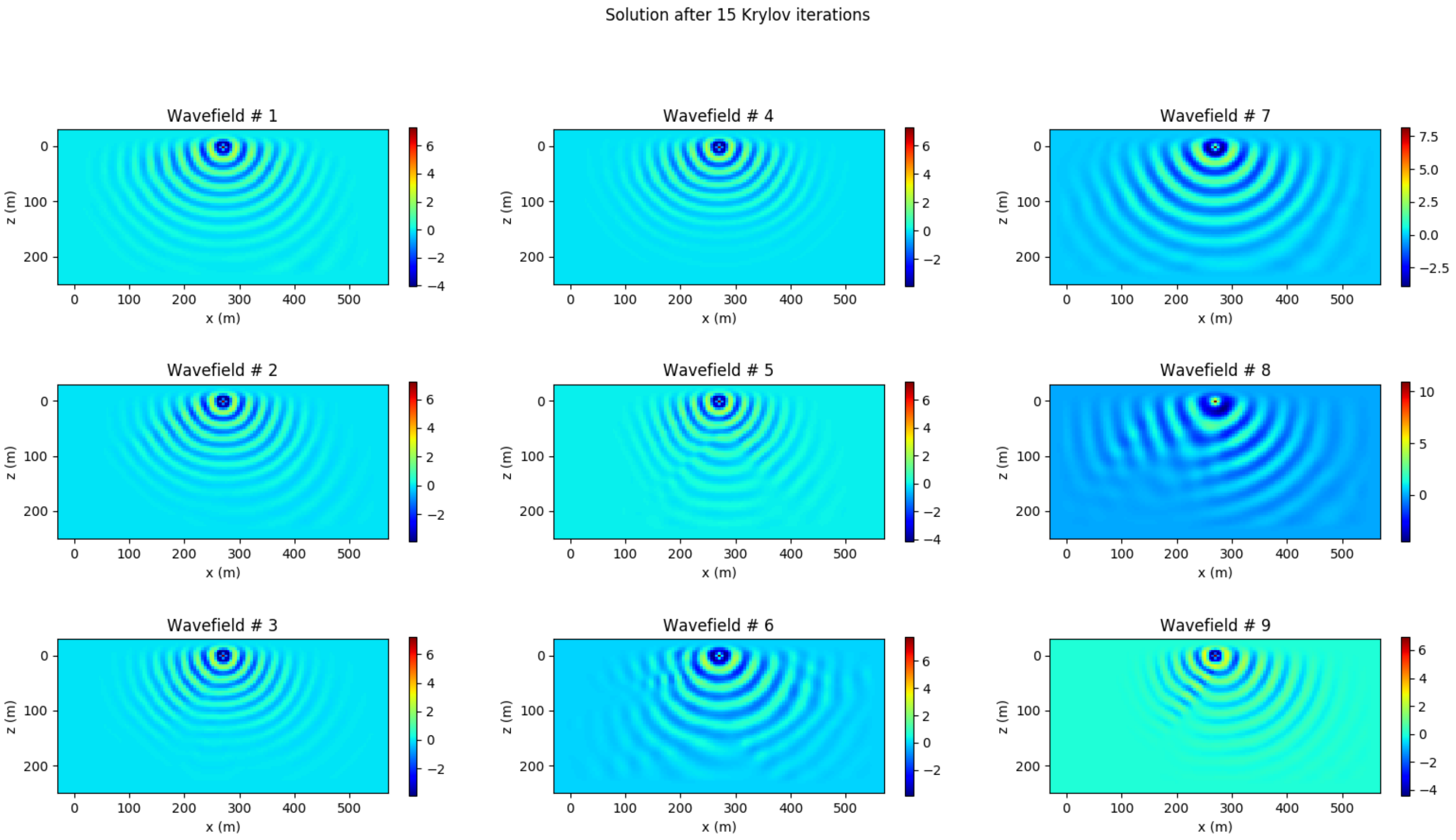
Example 2: solution wavefield at 60 Hz (after 5 Krylov iterations)



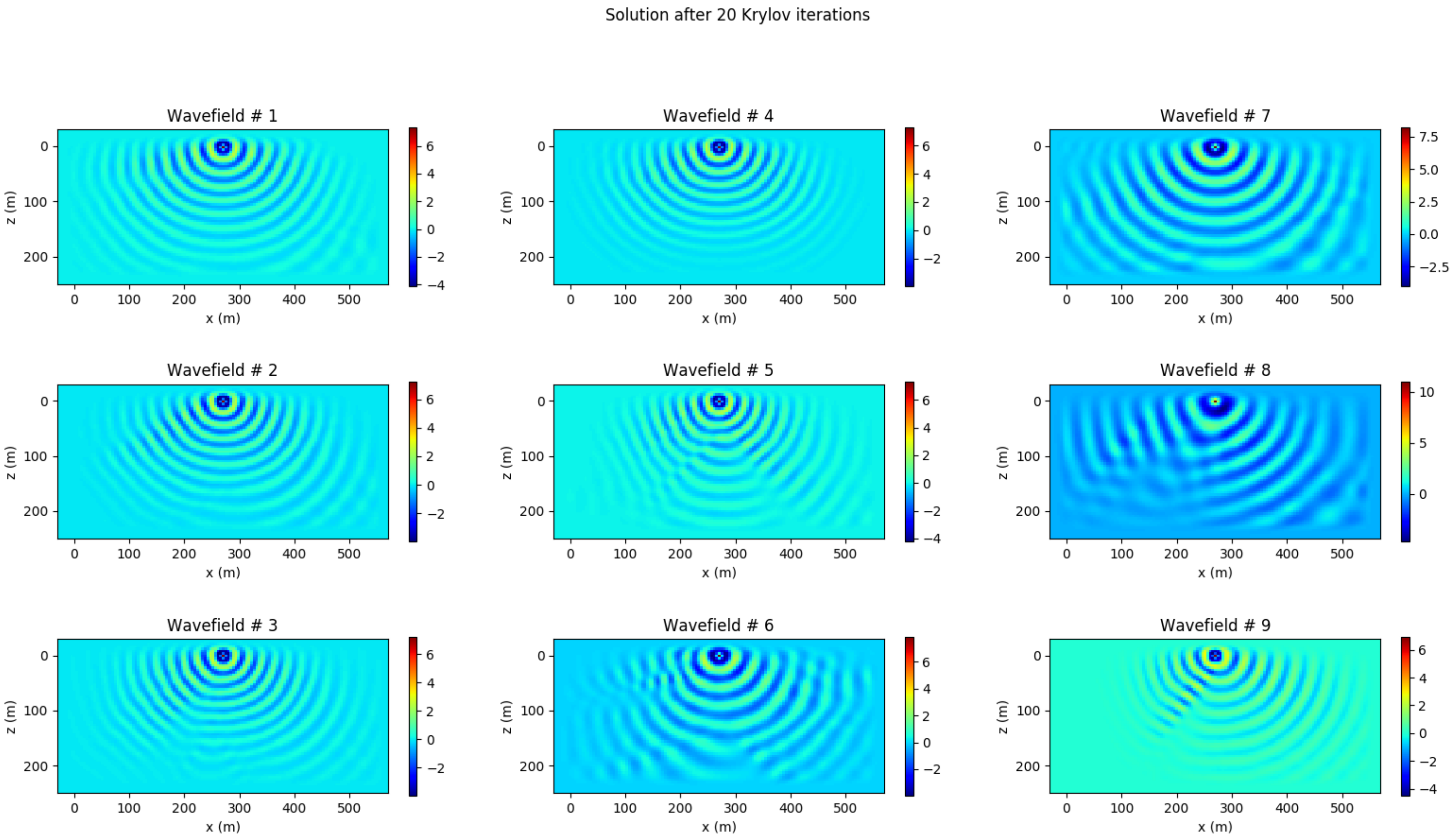
Example 2: solution wavefield at 60 Hz (after 10 Krylov iterations)



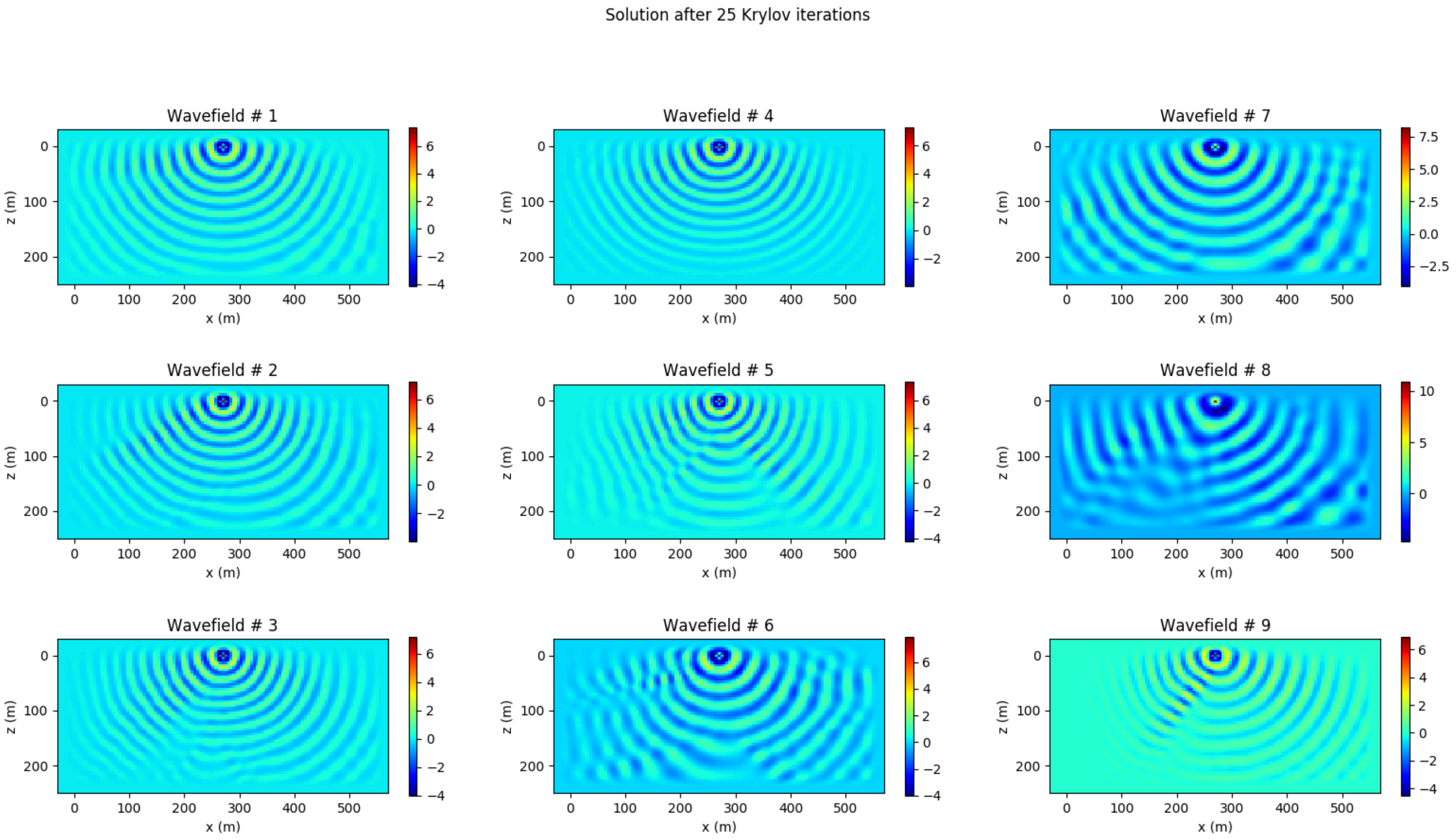
Example 2: solution wavefield at 60 Hz (after 15 Krylov iterations)



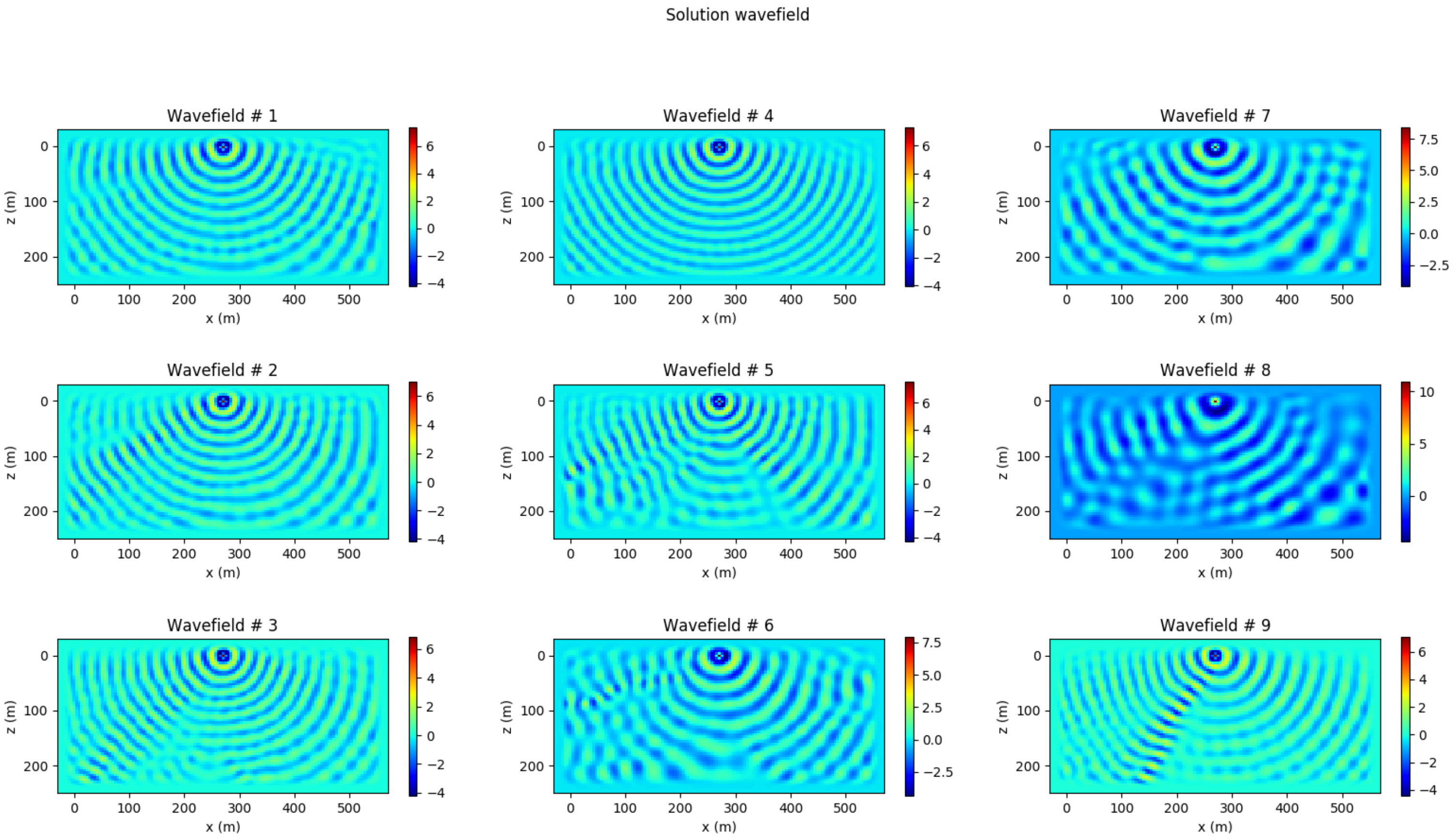
Example 2: solution wavefield at 60 Hz (after 20 Krylov iterations)



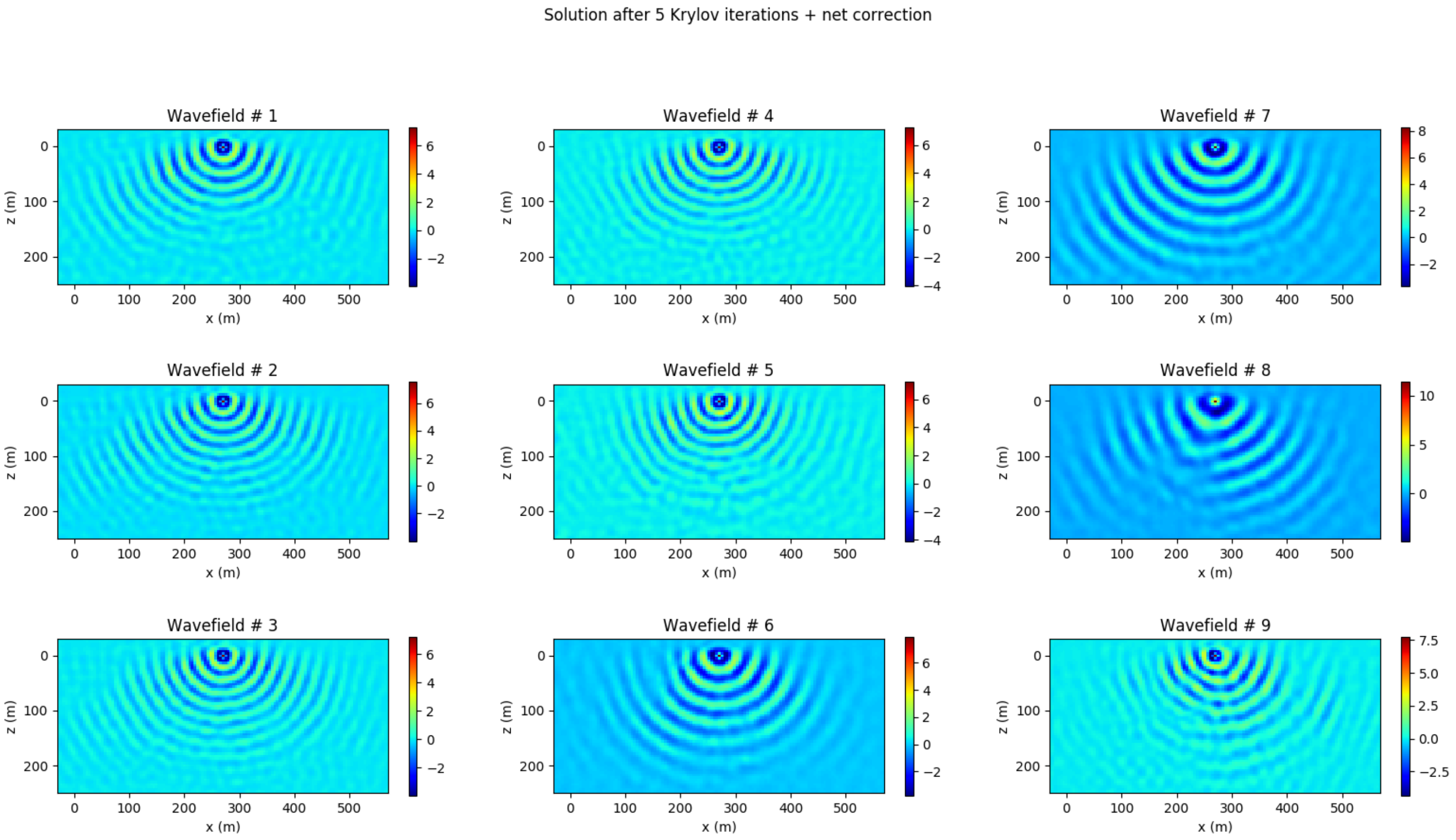
Example 2: solution wavefield at 60 Hz (after 25 Krylov iterations)



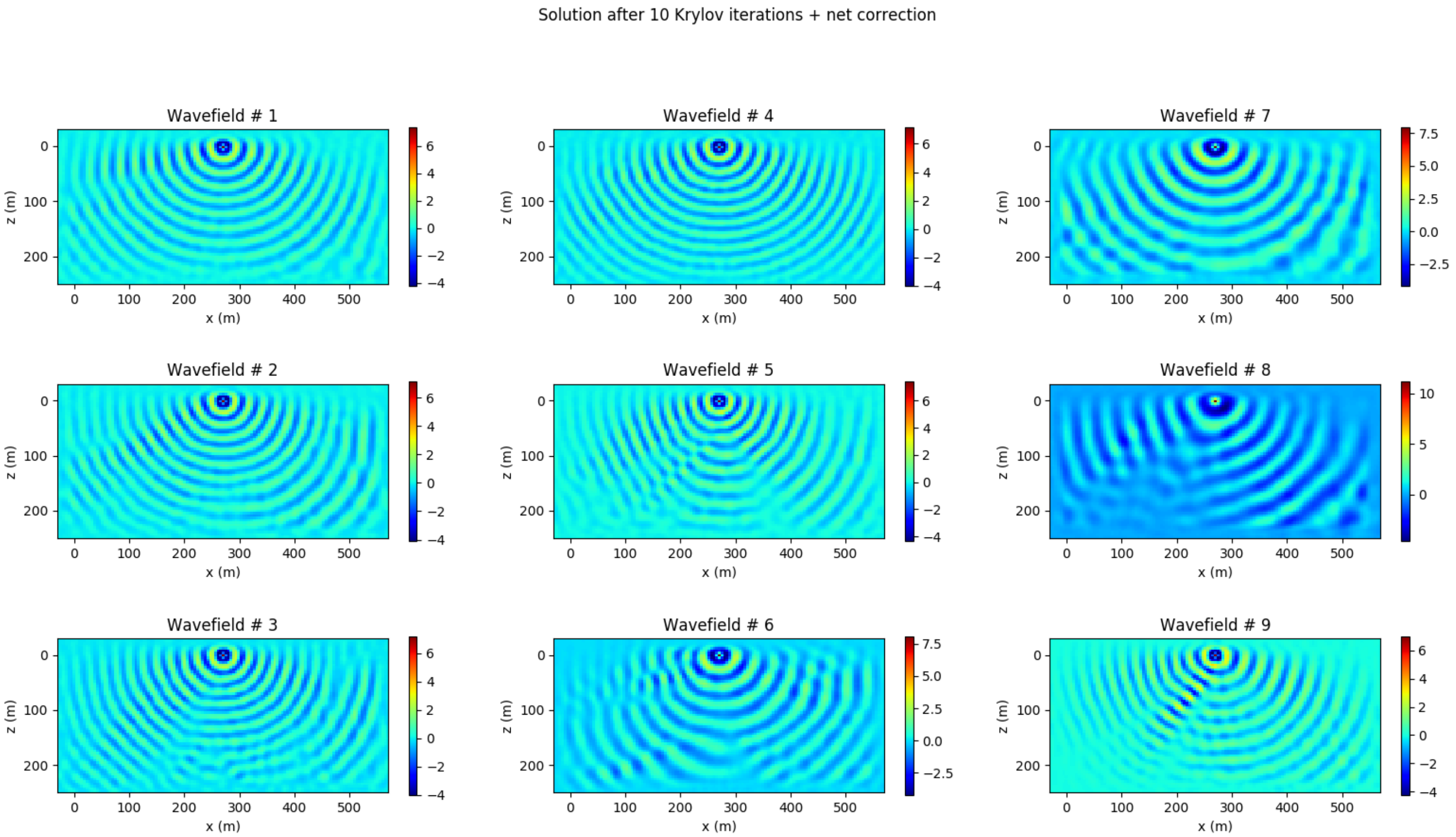
Example 2: solution distribution at 60 Hz (train size: 1024, test size: 16)



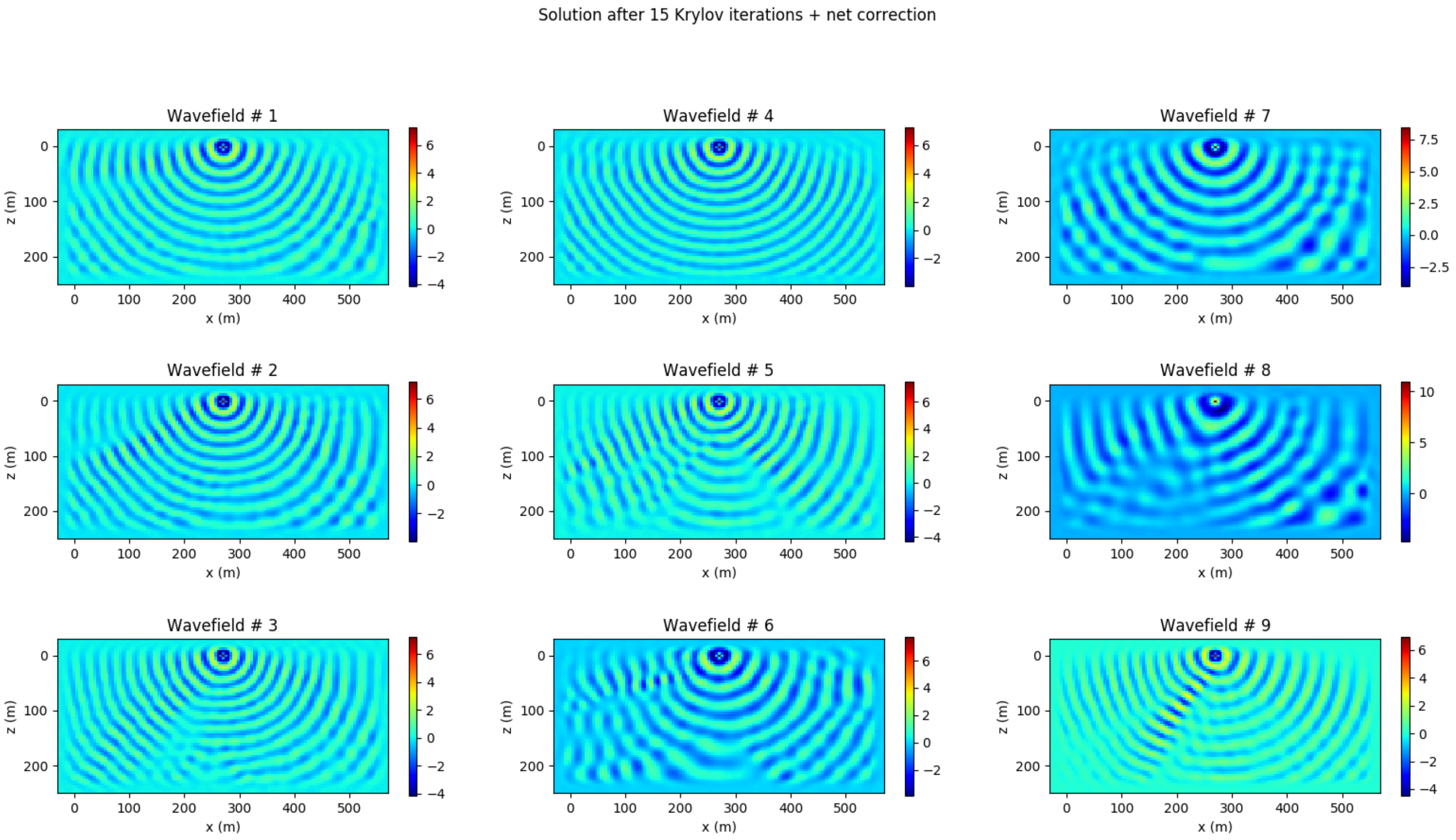
Example 2: solution wavefield at 60 Hz (after 5 Krylov iterations **+ net**)



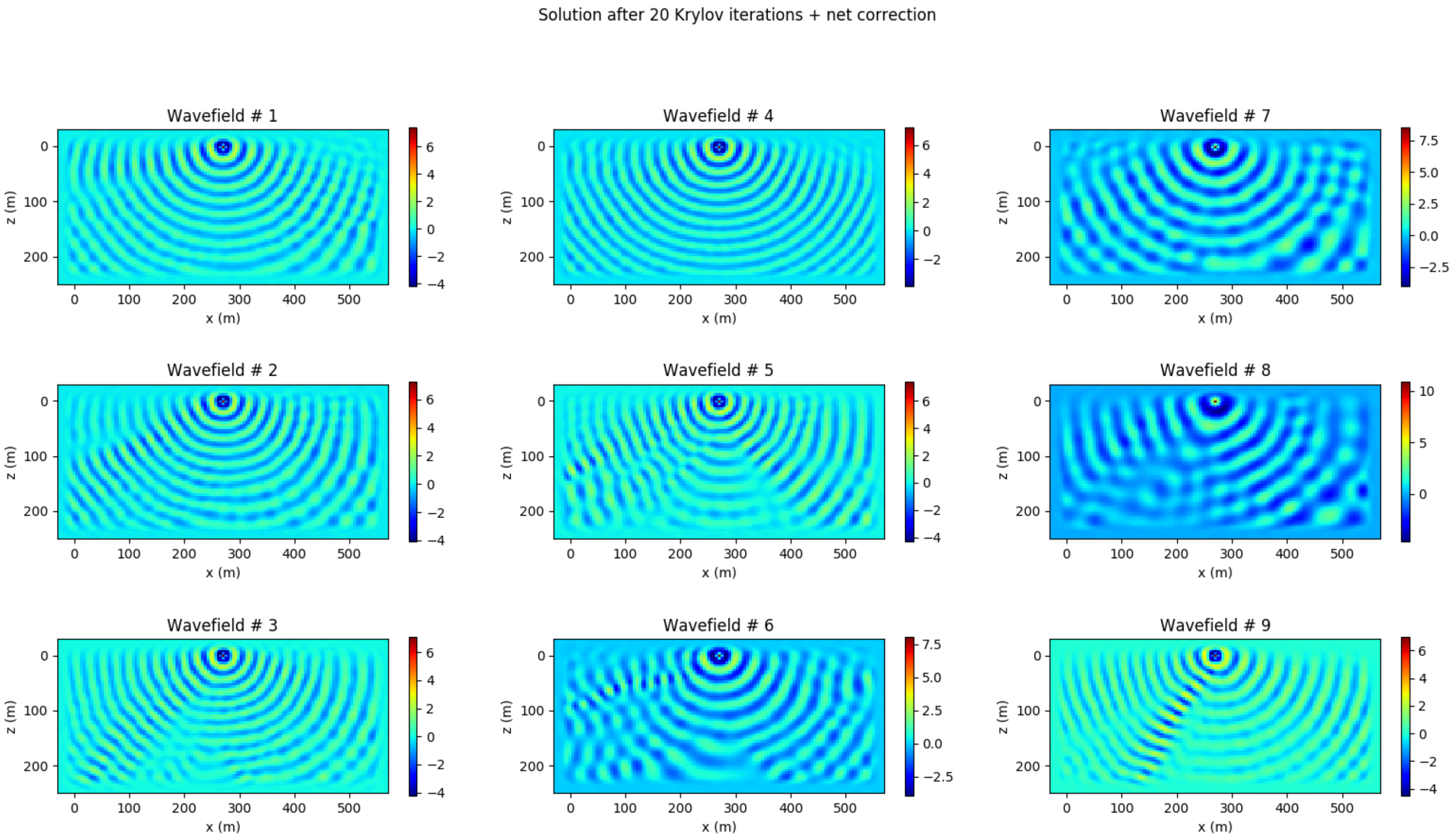
Example 2: solution wavefield at 60 Hz (after 10 Krylov iterations **+ net**)



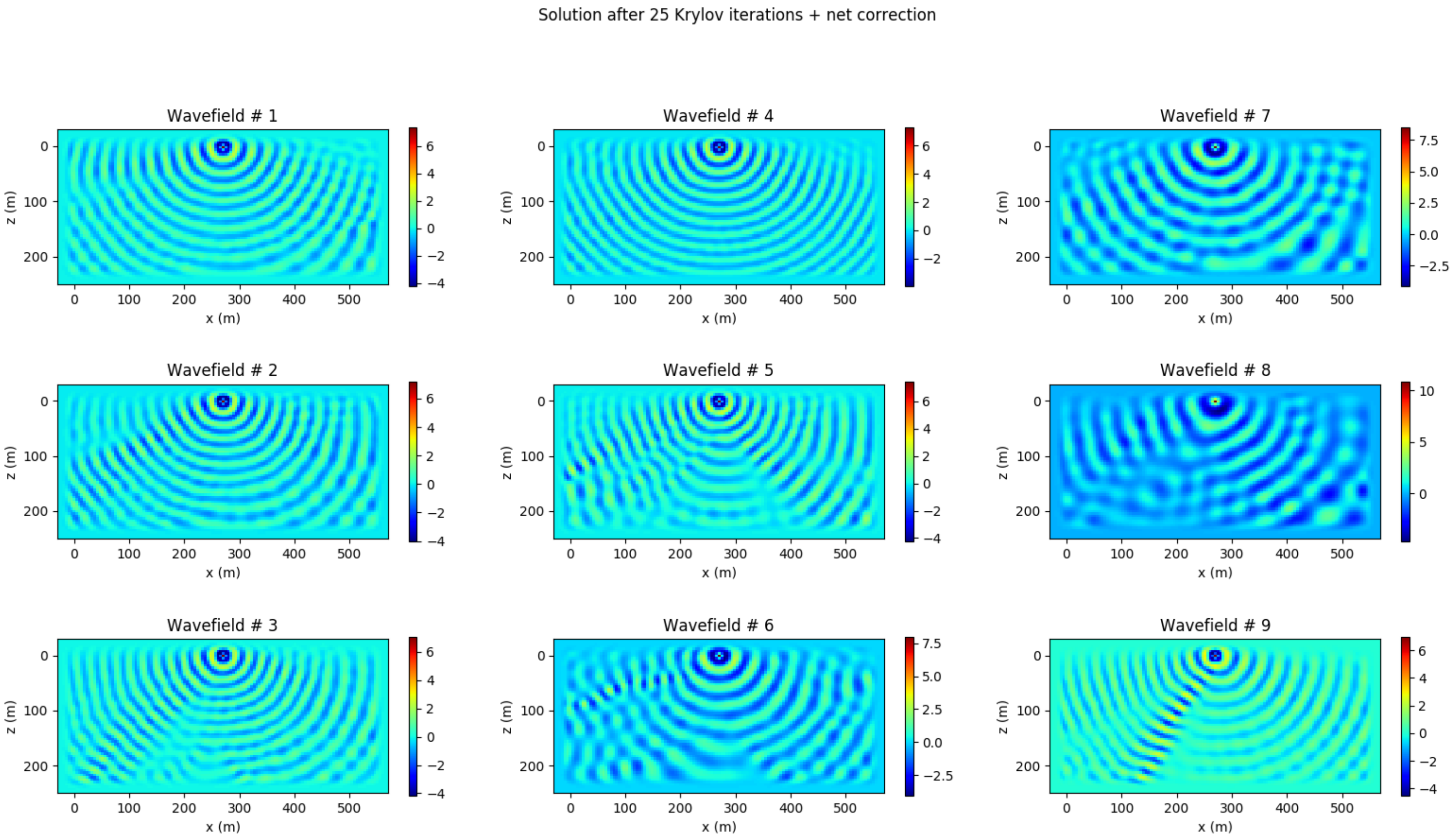
Example 2: solution wavefield at 60 Hz (after 15 Krylov iterations + net)



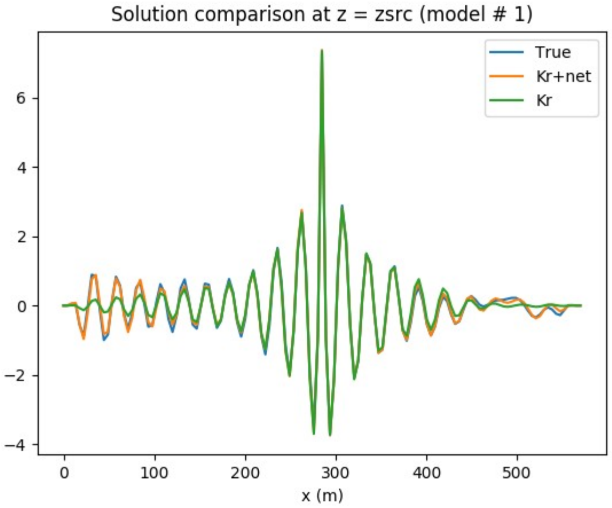
Example 2: solution wavefield at 60 Hz (after 20 Krylov iterations **+ net**)

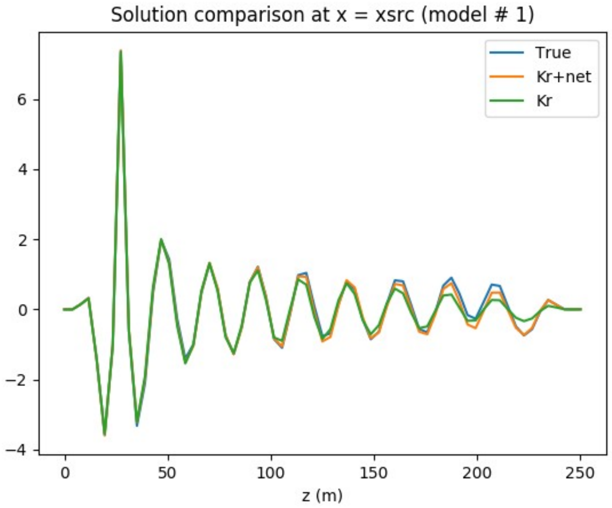


Example 2: solution wavefield at 60 Hz (after 25 Krylov iterations **+ net**)

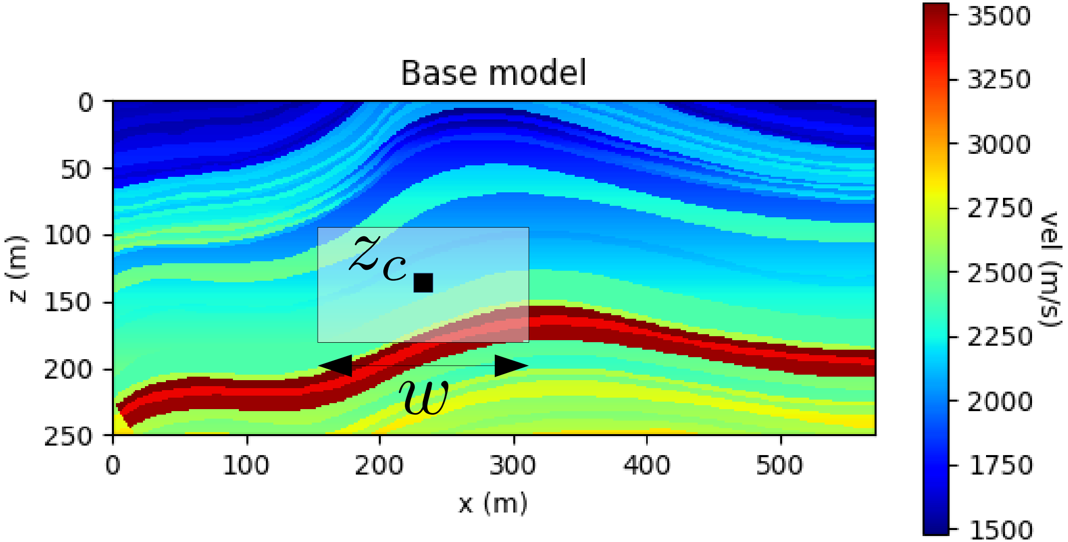


Example 2: solution trace comparison





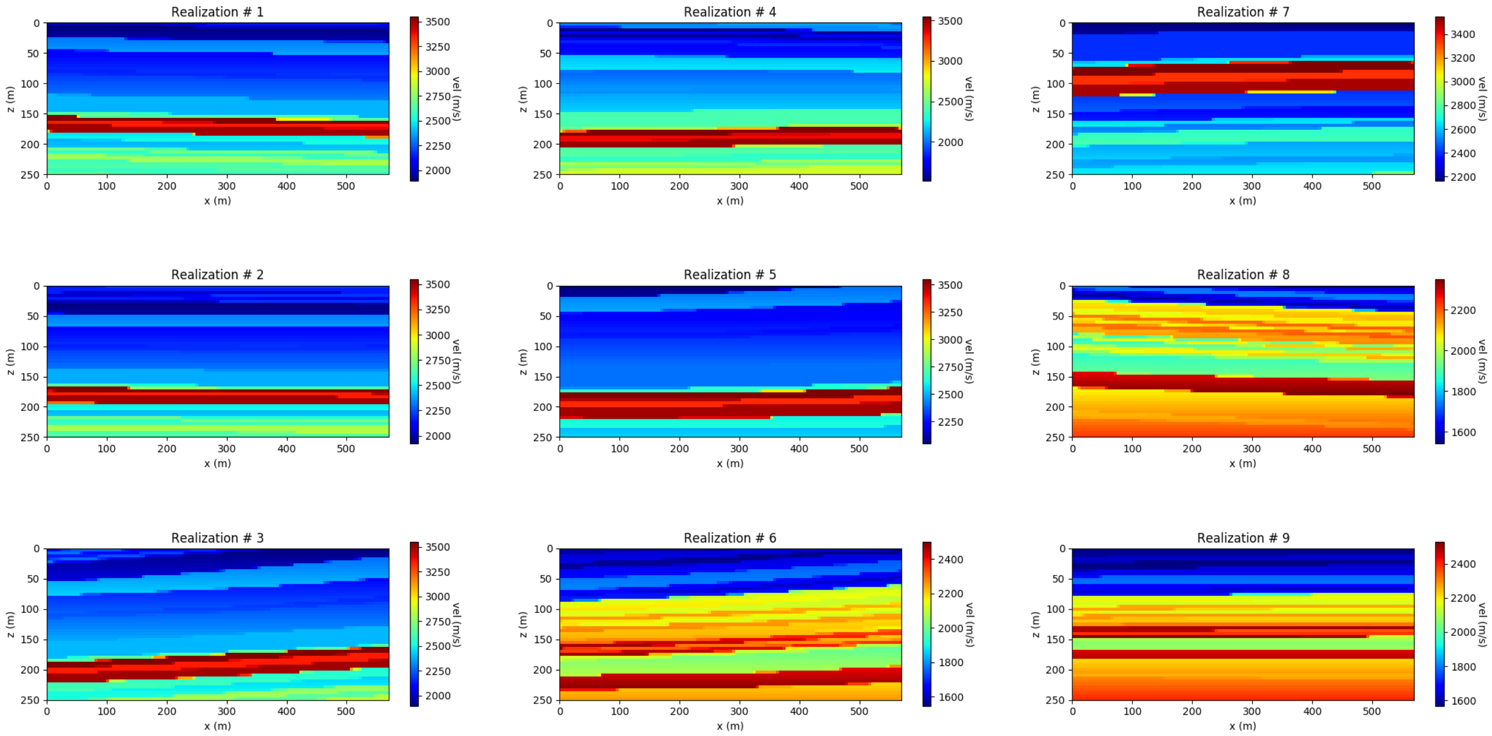
Example 2: testing generalization to different model distributions



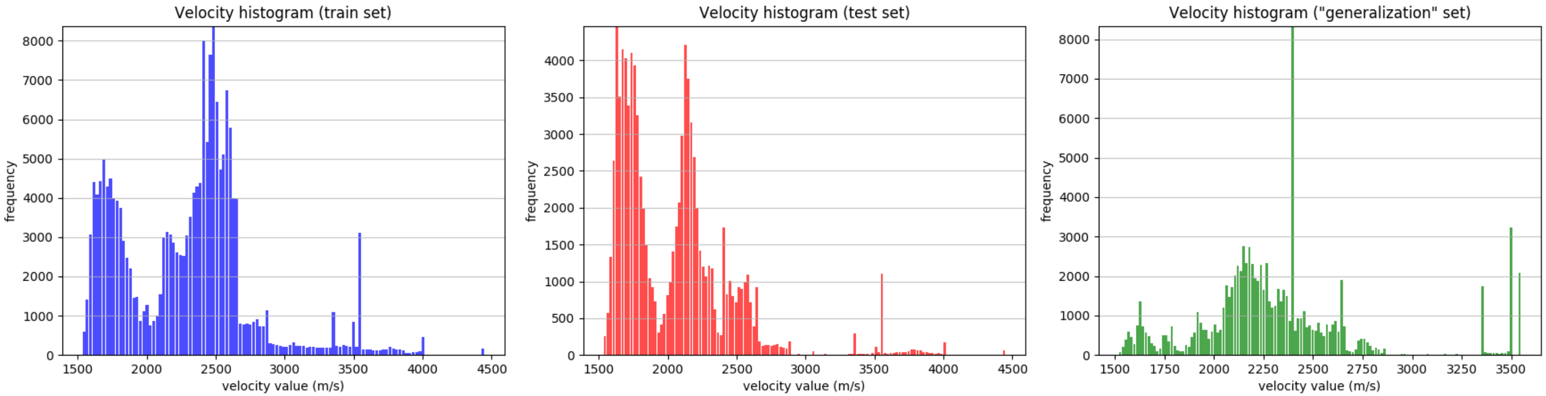
$$z_c \sim U(z_0, z_{\text{end}}) \qquad w \sim w_0 + U(-a, a)$$

Example 2: Marmousi-like distribution (generalization test size: 16)

Generalization test set excerpt



Example 2: Train/test errors



$$L = \sqrt{\sum_i ||\mathbf{u}_i^{\mathrm{sol}} - \tilde{\mathbf{u}}_i||^2 / \sum_i ||\mathbf{u}_i^{\mathrm{sol}}||^2}$$

Final loss	Train error	Test error	“Generalization” error
Krylov iterations	37.6%	40.5%	32.3%
Krylov net	12.1%	12.9%	17.2%

Conclusions/Future plans

Possible improvements:

- GANs

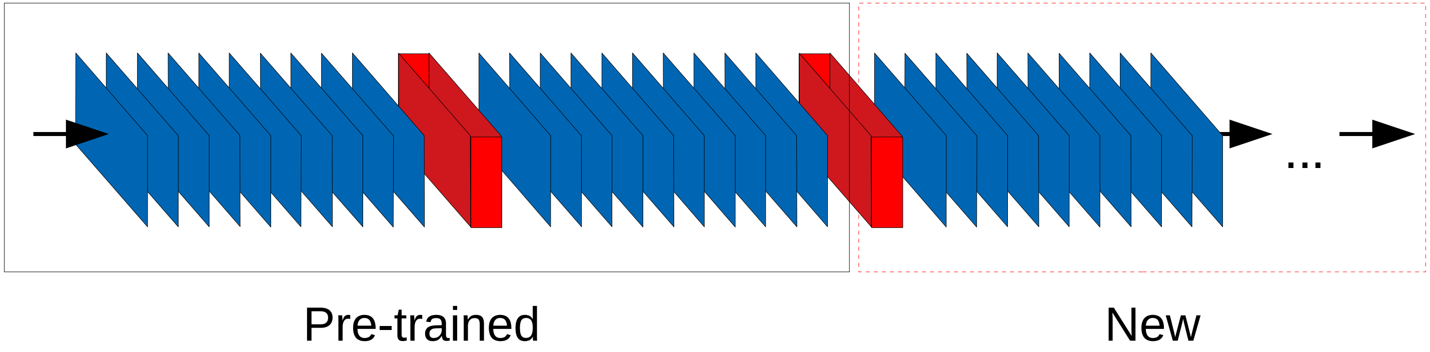
$D_\varphi : \mathcal{U} \to [0,1]$ discriminator

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

Conclusions/Future plans

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

e.g., smoothing: $\mathbf{x}^h \leftarrow \mathbf{x}^h + N_\theta^h(\mathbf{r}^h)$

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

$$L = \sum_j ||R_h^{jh}(\mathbf{f} - H[\mathbf{m}]\, F_\theta(\mathbf{m}))||^2$$

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

References (1/2)

Adler, J., and O. Öktem, Solving ill-posed inverse problems using iterative deep neural networks, Inverse Problems (2017)

Chu, M., and N. Thuerey, Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors, ACM Transactions on Graphics (2017)

Erlangga, Y. A., C. W. Oosterlee, and C. Vuik, A novel multigrid based preconditioner for the heterogeneous Helmholtz problems, SIAM J. Sci. Comput. (2006)

Farimani, A. B., J. Gomes, and V. Pande, Deep Learning the Physics of Transport Phenomena, arXiv preprint (2017)

George, A., Nested Dissection of a Regular Finite Element Mesh, SIAM J. Numer. Anal. (1973)

Guo, X., W. Li, and F. Iorio, Convolutional Neural Networks for Steady Flow Approximation, KDD (2016)

Haber, E., and L. Ruthotto, Stable architectures for deep neural networks, Inverse Problems (2017)

He, J., and J. Xu, MgNet: A Unified Framework of Multigrid and Convolutional Neural Network, arXiv preprint (2019)

He, K., X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition. arXiv preprint (2015)

Hsieh, J.-T., S. Zhao, S. Eismann, L. Mirabella, and S. Ermon, Learning neural PDE solvers with convergence guarantees, ICLR (2019)

Ke, T.-W., M. Maire, and S. X. Xu, Multigrid Neural Architectures, CVPR (2017)

Kingma, D. P., and Ba, J. L., ADAM: a method for stochastic optimization, ICLR (2015)

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)

Krizhevsky, A., and G. Hinton, Learning multiple layers of features from tiny images, Technical report (2009)

Kutz, N., Deep learning in fluid dynamics, J. Fluid Mech. (2017)

References (2/2)

Mills, K., M. Spanner, and I. Tamblyn, Deep learning and the Schrödinger equation, Phys. Rev. A (2017)

Mulder, W., and R.-E. Plessix, Time- versus frequency-domain modelling of seismic wave propagation, EAGE abstract (2002)

Saad, Iterative methods for sparse linear systems, SIAM (2003)

Sharma, R., A. B. Farimani, J. Gomes, P. Eastman, and V. Pande, Weakly-Supervised Deep Learning of Heat Transport via Physics Informed Loss, arXiv preprint (2018)

Siahkoohi, A., M. Louboutin, R. Kumar, and F. J. Herrmann, “Deep Convolutional Neural Networks in prestack seismic–-two exploratory examples”, SEG abstract (2018)

Sirignano, J., and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, arXiv preprint (2018)

Singh, A. P., Sh. Medida, and K. Duraisamy, Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils, AIAA J. (2017)

Tang, W., T. Shan, X. Dang, M. Li, F. Yang, S. Xu, and J. Wu, Study on a Poisson’s Equation Solver Based On Deep Learning Technique, EDAPS (2017)

Tompson, J., K. Schlachter, P. Sprechmann, and K. Perlin, Accelerating Eulerian Fluid Simulation With Convolutional Networks, PMLR (2017)

Yang, C., X. Yang, and X. Xiao, Data-driven projection method in fluid simulation, Comp. Anim. Virtual Worlds (2016)

Zhang, Z., L. Zhang, Z. Sun, N. Erickson, R. From, and J. Fan, Solving Poisson’s Equation using Deep Learning in Particle Simulation of PN Junction, arXiv preprint (2018)