Multi-parameter waveform inversion; exploiting the structure of penalty-methods

Bas Peters

June, 2014

SLIM
University of British Columbia

Outline

- Challenging problem example (same as I presented on Tuesday) + extra details
- Waveform Reconstruction Inversion based multi-parameter algorithms

Wavefield Reconstruction Inversion [T. van Leeuwen & F.J. Herrmann, 2013]

Objective:

$$\bar{\phi}_\lambda(\mathbf{m}) = \frac{1}{2}\sum_{kl} \underbrace{\|P\bar{\mathbf{u}}_{kl} - \mathbf{d}_{kl}\|_2^2}_{\text{Data-misfit}} + \frac{\lambda^2}{2} \underbrace{\|H_k(\mathbf{m})\bar{\mathbf{u}}_{kl} - \mathbf{q}_{kl}\|_2^2}_{\text{PDE-misfit}}$$

where $\bar{\mathbf{u}}_{kl} = \arg\min_{\mathbf{u}_{kl}} \left\| \begin{pmatrix} \lambda H_k(\mathbf{m}) \\ P \end{pmatrix} \mathbf{u}_{kl} - \begin{pmatrix} \lambda \mathbf{q}_{kl} \\ \mathbf{d}_{kl} \end{pmatrix} \right\|_2$

and λ is a tradeoff parameter between PDE-fit and data-fit

Wavefield Reconstruction Inversion [T. van Leeuwen & F.J. Herrmann, 2013]

Objective:

Data-misfit

PDE-misfit

$$\bar{\phi}_\lambda(\mathbf{m}) = \frac{1}{2}\sum_{kl} \|P\bar{\mathbf{u}}_{kl} - \mathbf{d}_{kl}\|_2^2 + \frac{\lambda^2}{2}\|H_k(\mathbf{m})\bar{\mathbf{u}}_{kl} - \mathbf{q}_{kl}\|_2^2$$

with gradient:

$$\nabla_{\mathbf{m}}\bar{\phi}_\lambda = \sum_{kl} \lambda^2 G_{kl}(\mathbf{m}, \bar{\mathbf{u}}_{kl})^* \left(H_k(\mathbf{m})\bar{\mathbf{u}}_{kl} - \mathbf{q}_{kl}\right)$$

Non-linear waveform inversion

Example 1 (difficult):

- Lots of low frequencies missing, 24 frequency batches (15 iterations each) with intervals {5 6} ,{6 7},… ,{28 29} Hertz. Each interval contains 5 frequencies.
- We use 2 cycles through the batches: {5 6} ,{6 7},… ,{28 29}, {5 6} ,{6 7},… ,{28 29}
- Inaccurate initial model
- 103 sources and receivers near the surface, spread over the whole domain (6km).
- Source & receiver interval: 55m. Max. offset 6km.
- Shortest wavelength: 290m @ 5Hz. and 50m @ 29 Hz.
- Used Two-metric projection with L-BFGS Hessian for optimization with bound-constraints. **[Bertsekas, 1982 ; Gafni & Bertsekas, 1982 ; Schmidt, Kim & Sra, 2009]**

Why do we need more sophisticated algorithms?

- We need upper and lower bounds on the velocity for numerical reasons:
    - lower bound: sufficient number of grid points per wavelength
    - upper bound: very long wavelengths not absorbed by the PML
- Bounds can also incorporate a priori geological information
- Numerical experiments so far show that bound constraints are required for WRI.
- For pure gradient-descent, the model update can be projected onto the bounds (box).

Why do we need more sophisticated algorithms?

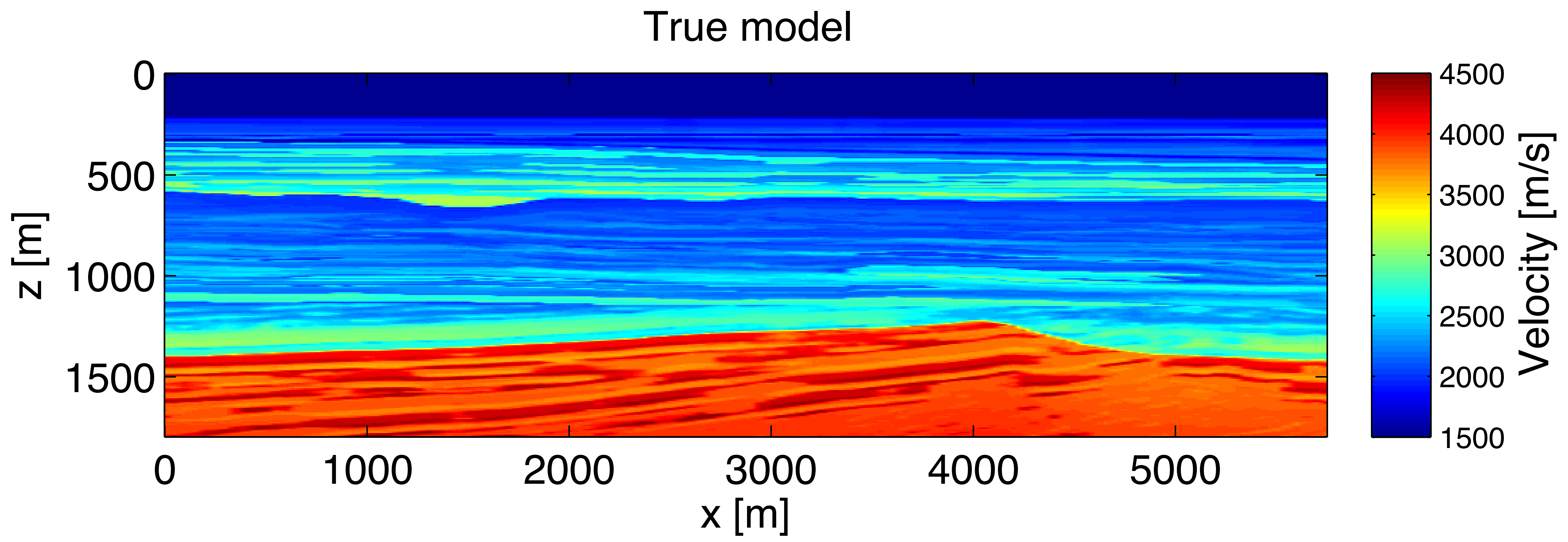
- Projecting l-BFGS/Newton updates onto the bounds is dangerous.
- The projected update may point in the opposite direction
- Projected-Newton methods exist, but can be computationally intensive.
- Multiple cheap solutions exist; the two-metric-projection (TMP) method is used here (with l-BFGS Hessian).
- TMP is a general concept, which can be used in conjunction with different Hessian approximations.

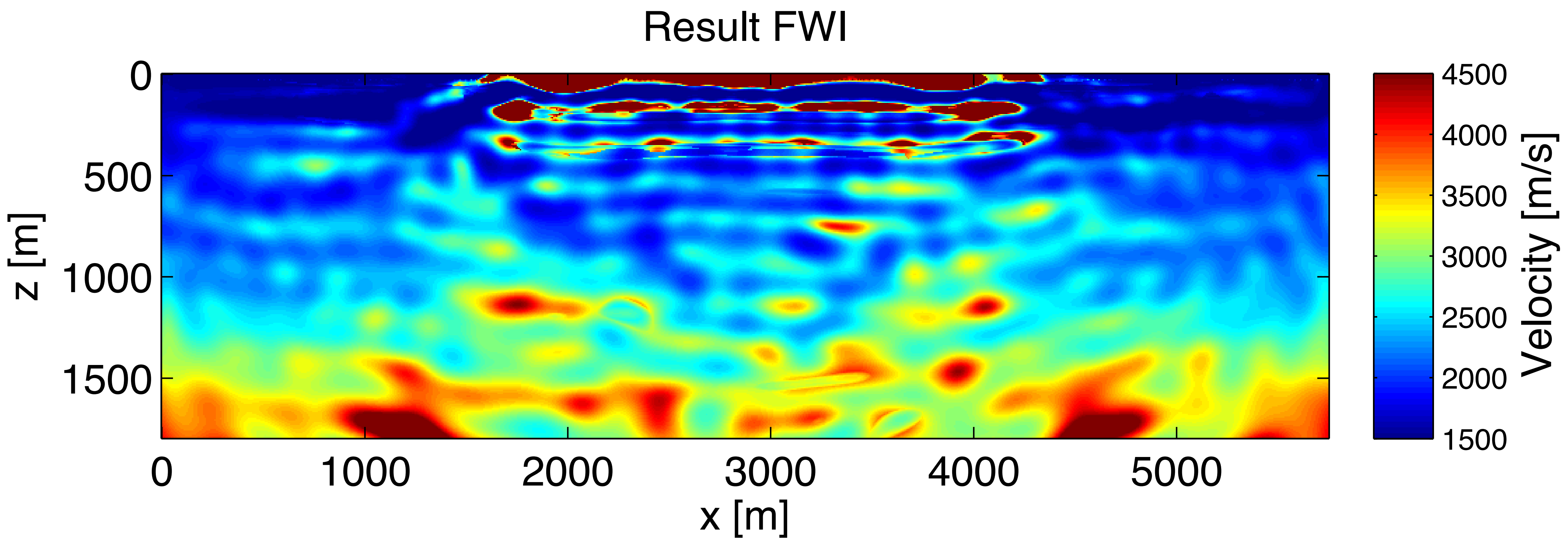
Two-metric-projection

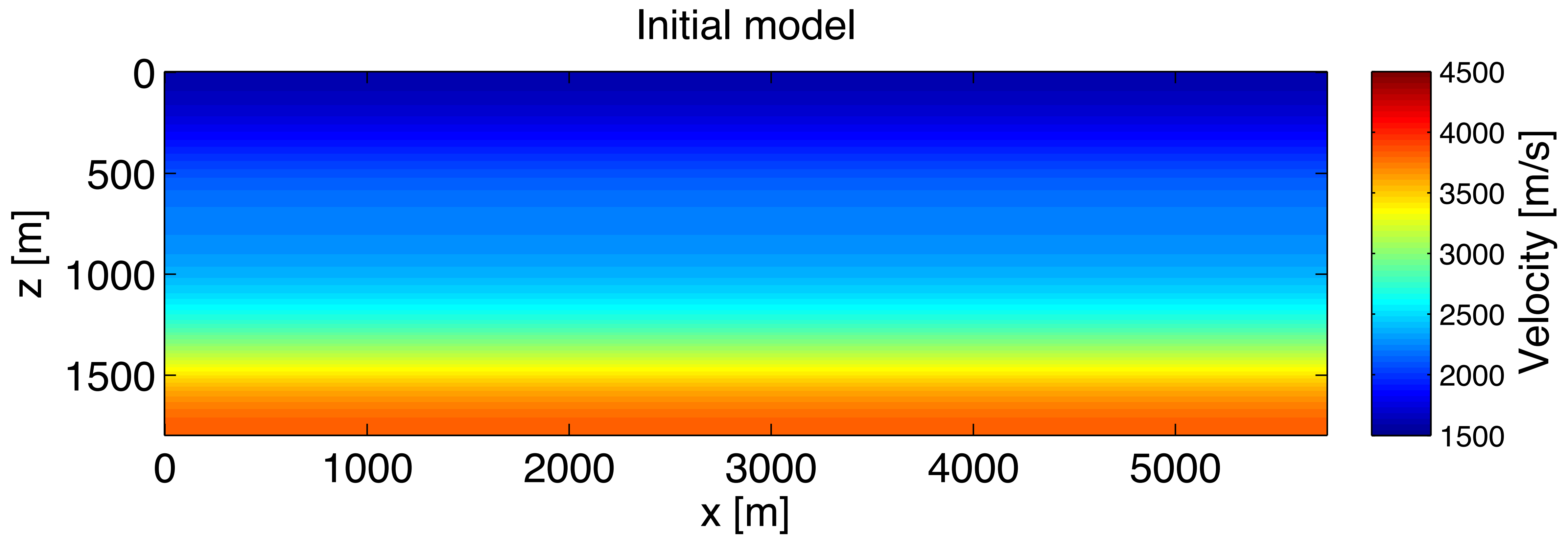
- Project in a different norm than the one used for computing the update direction
- Split Hessian approximation in free and restricted variables

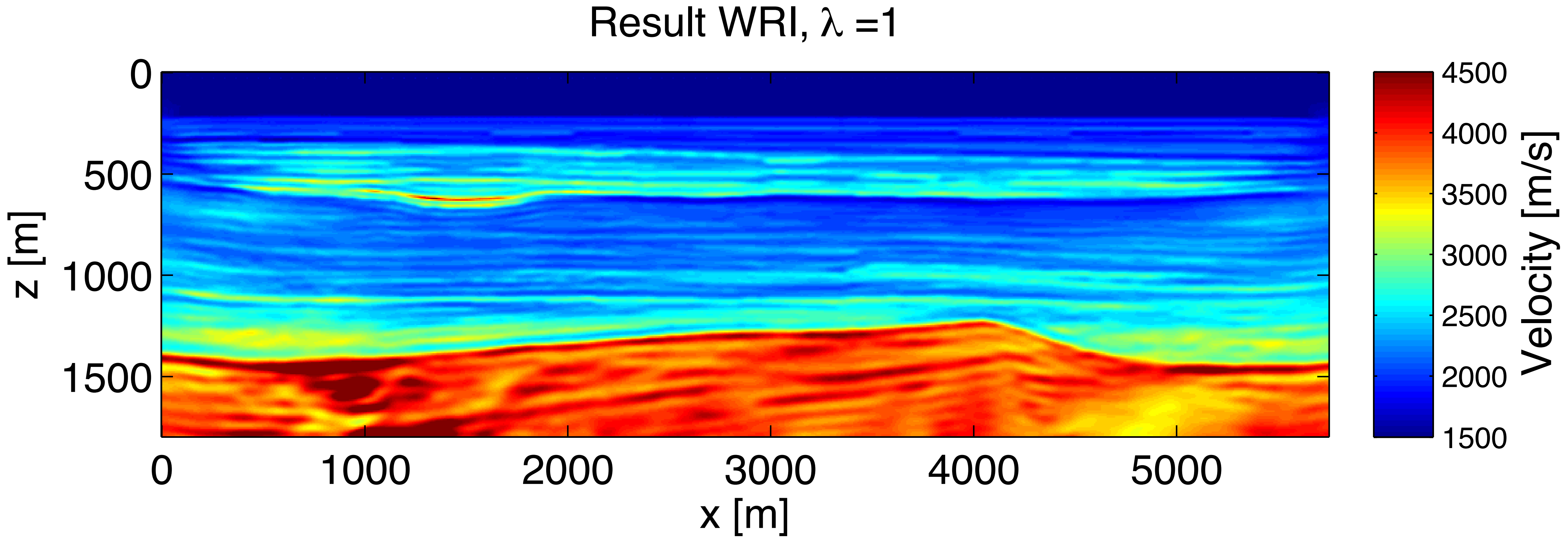
True, initial and final models

after 1st frequency batch



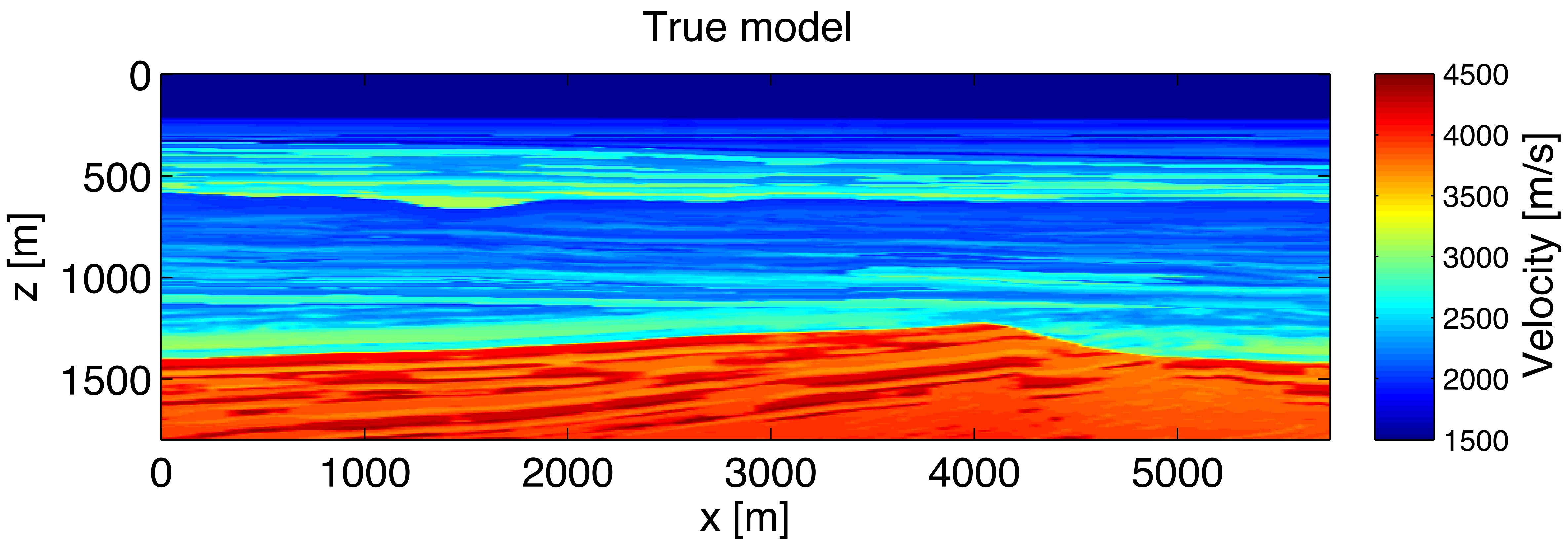




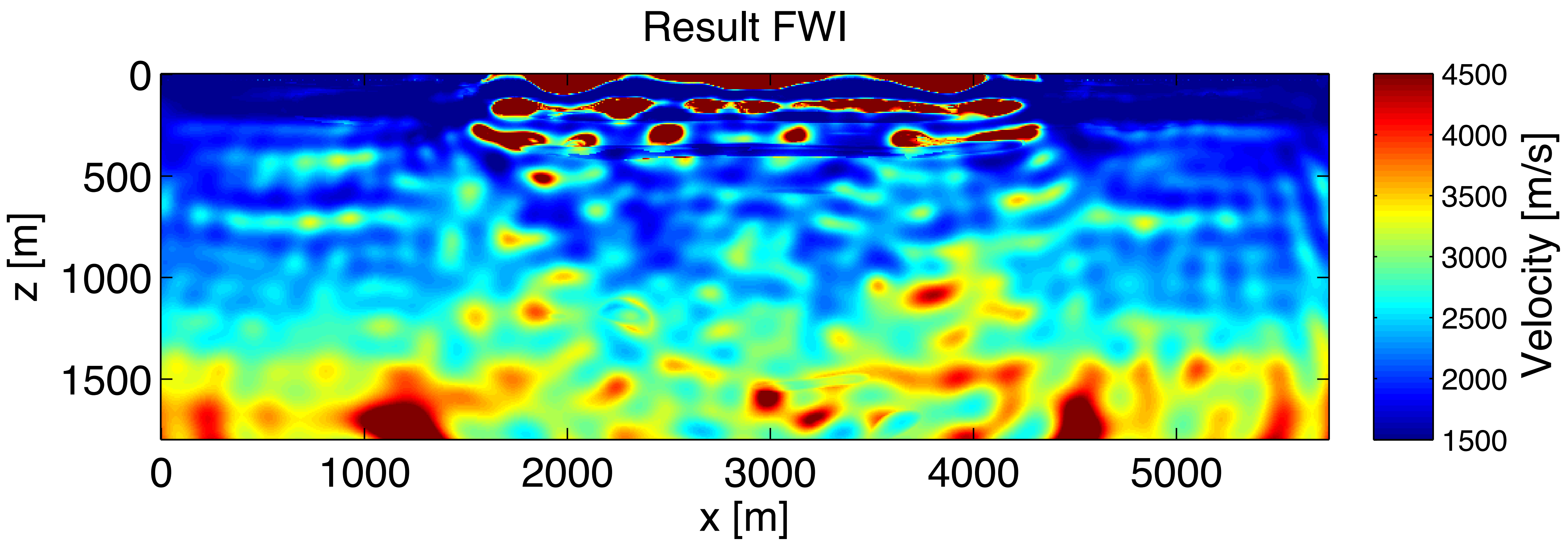


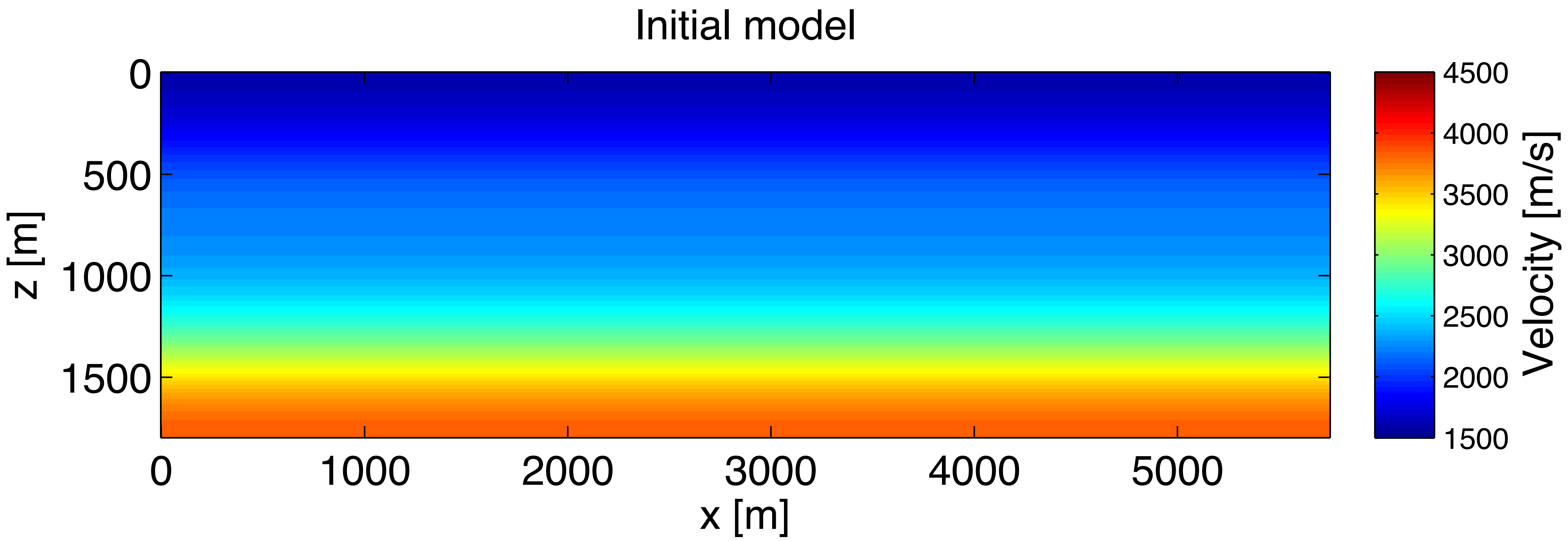
final result

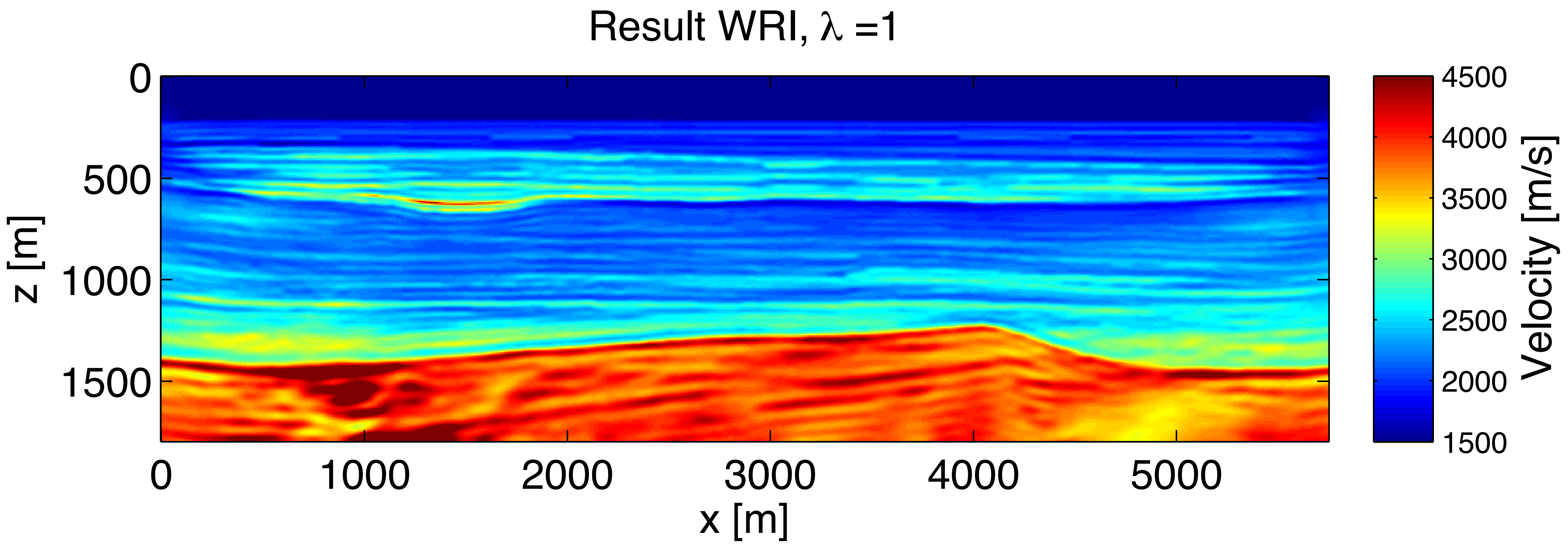
True, initial and final models



after 2nd frequency batch



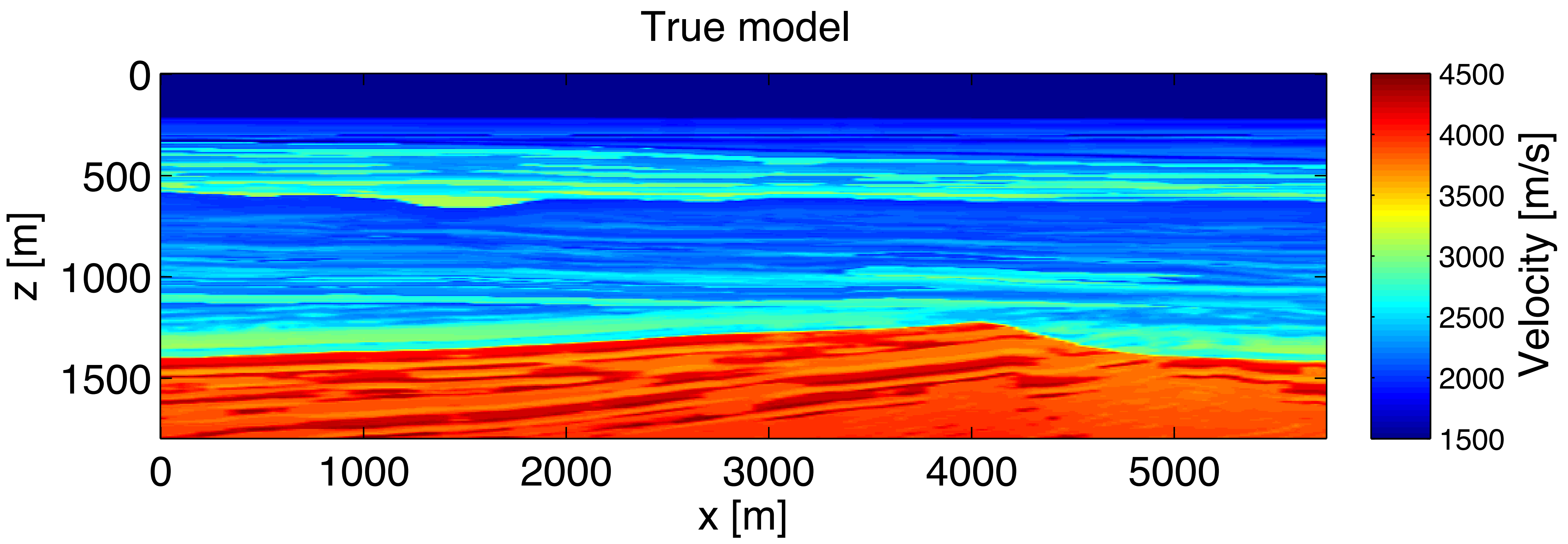


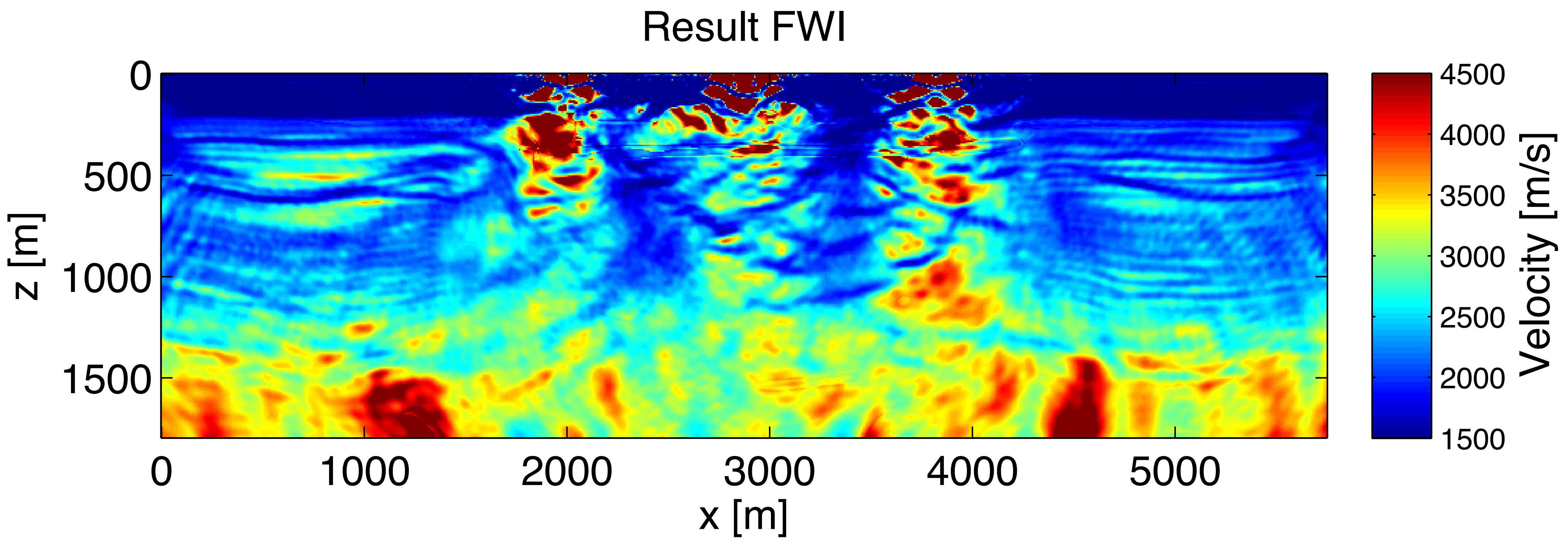


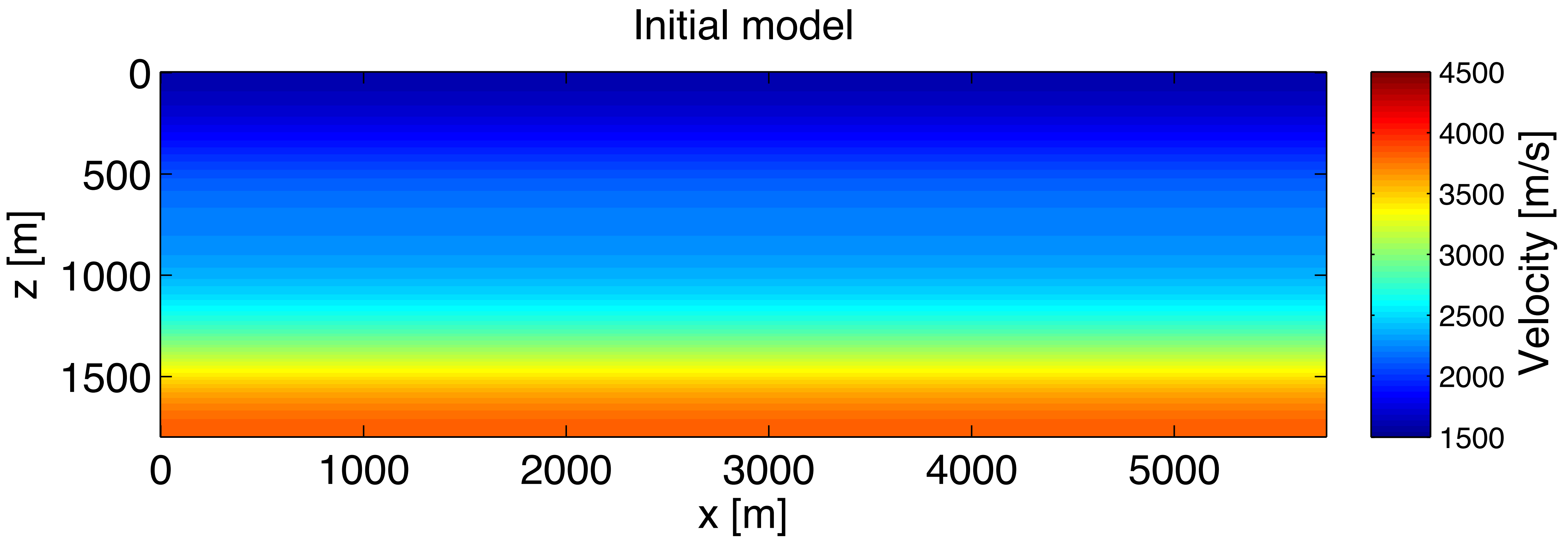
final result

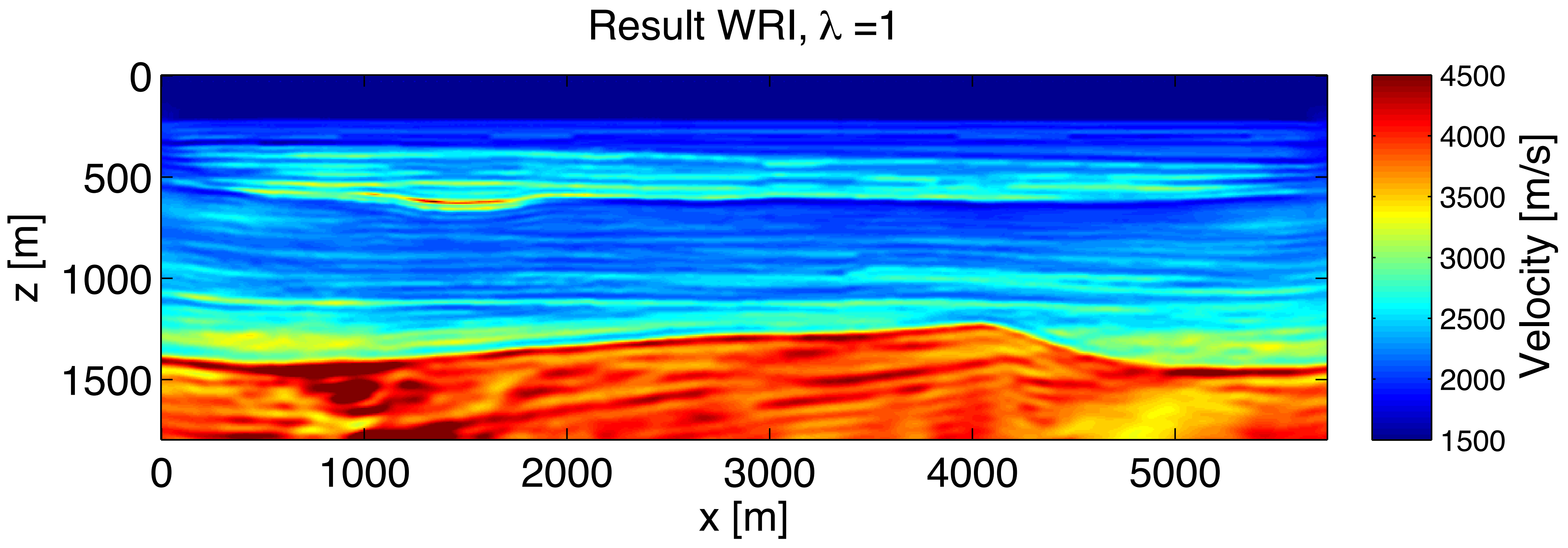
True, initial and final models

after last frequency batch



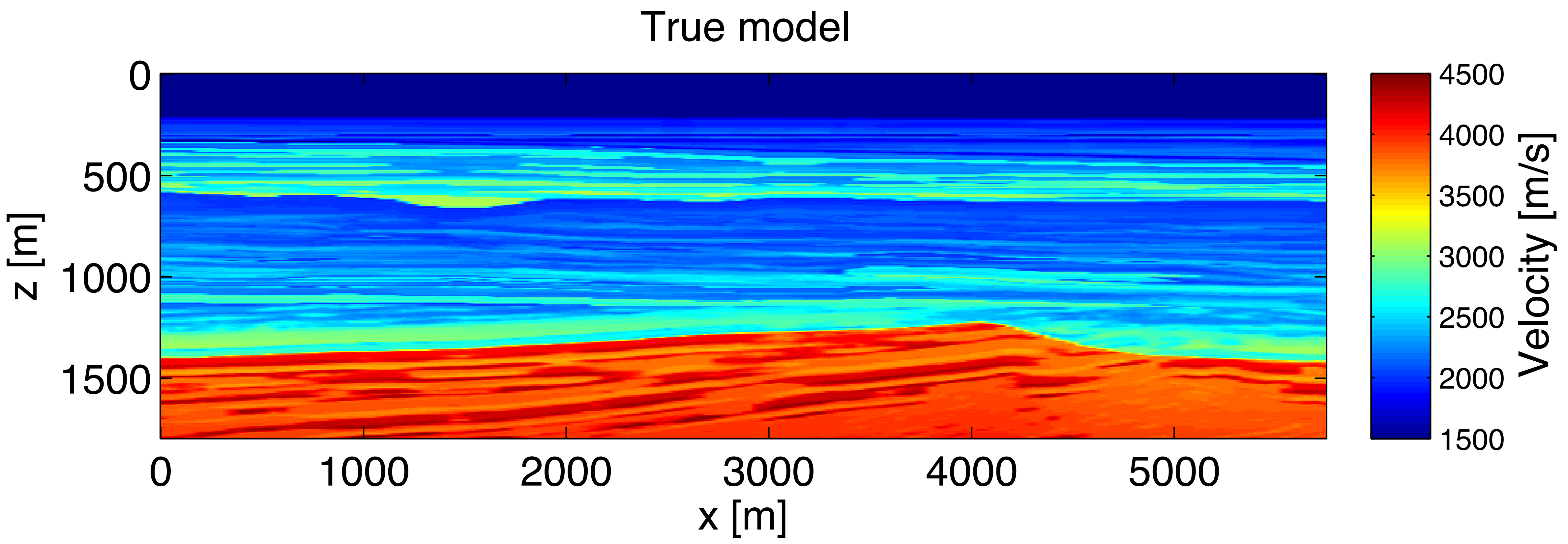


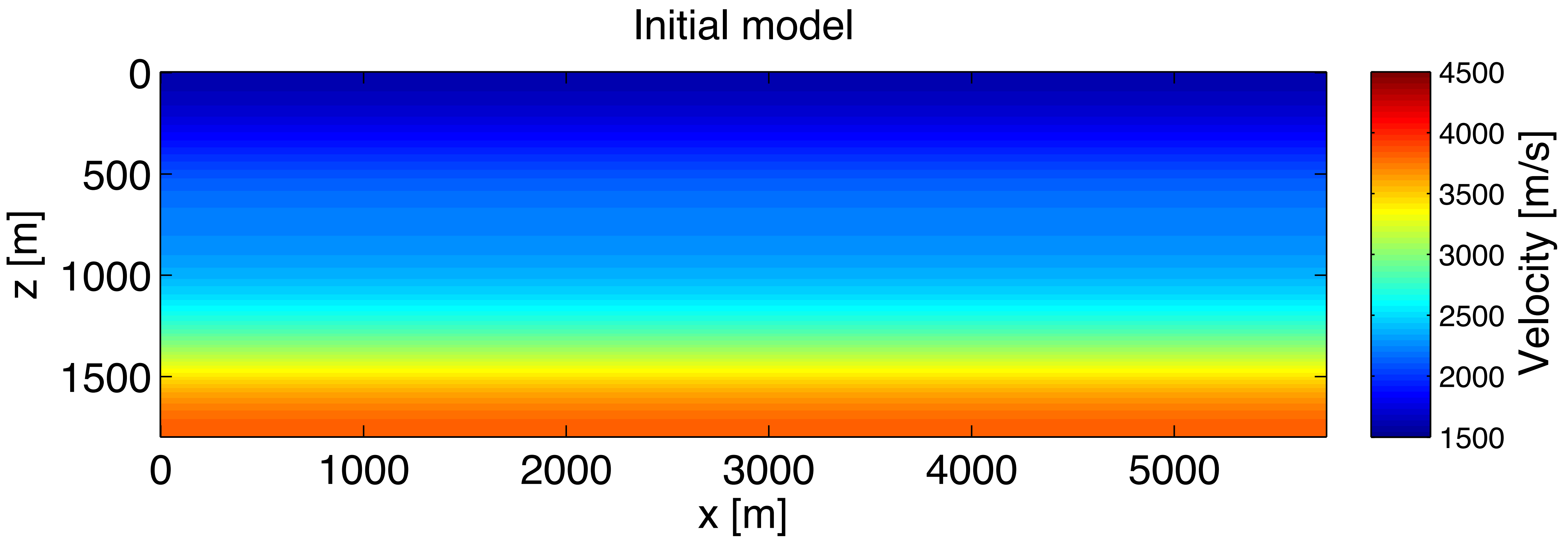




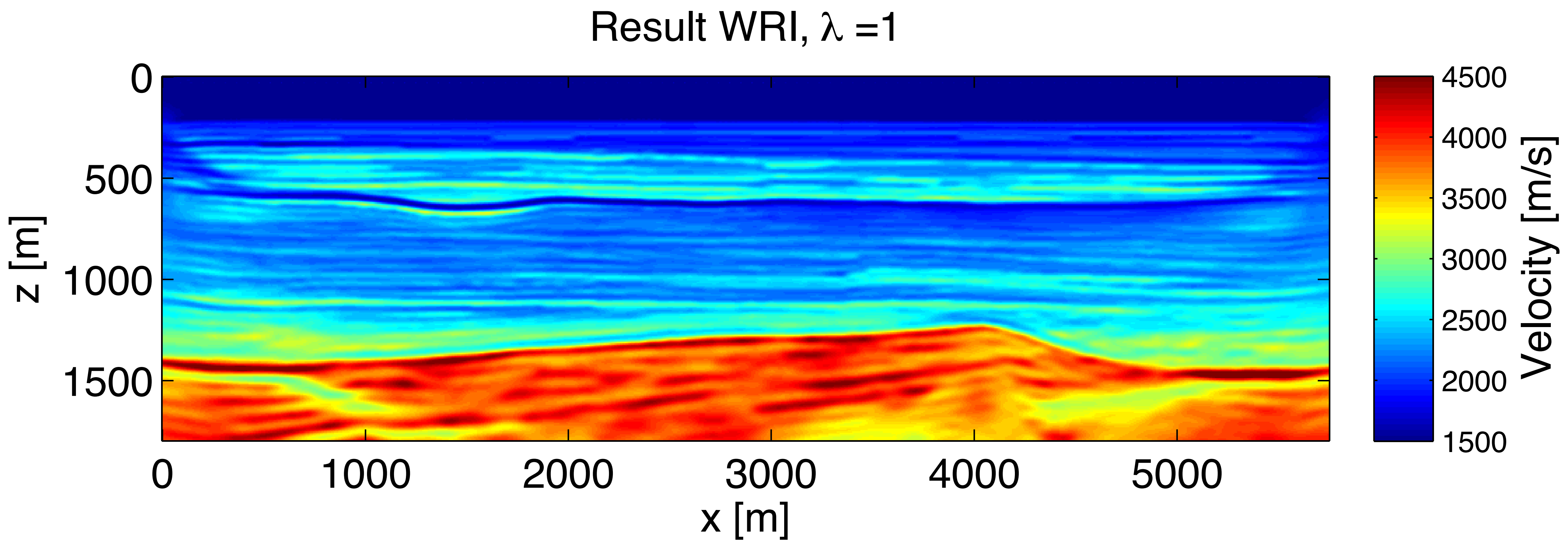
final result

True, initial and final models

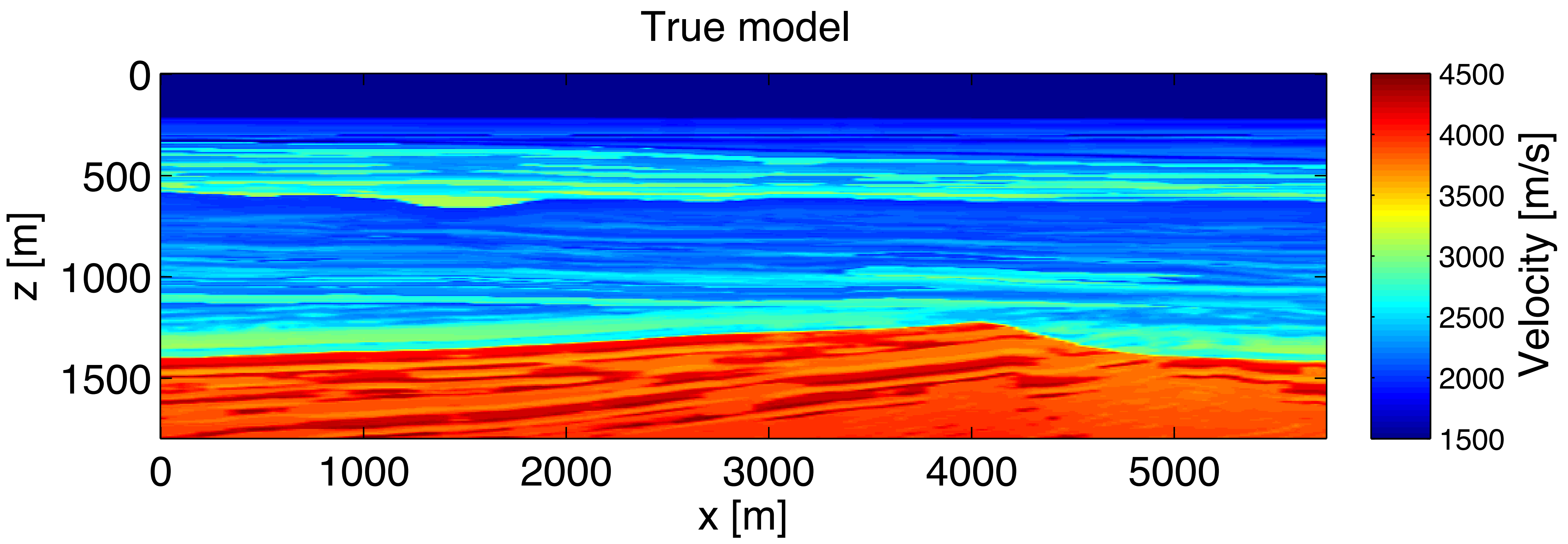


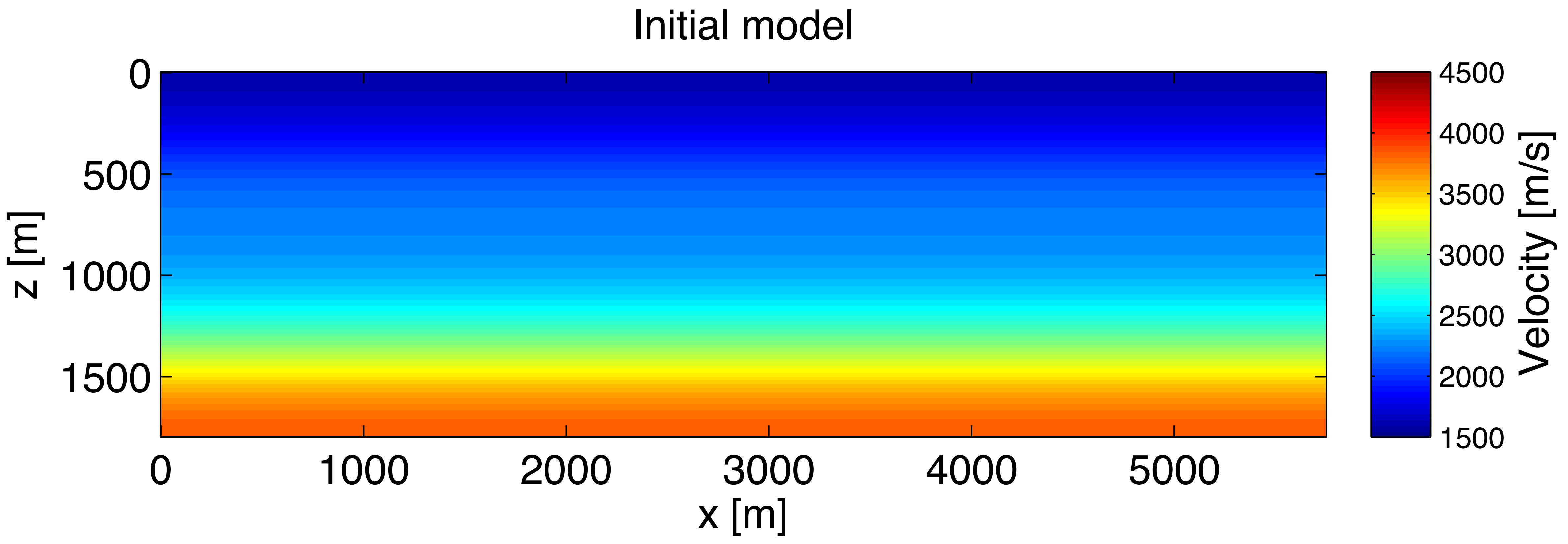


After 1st cycle

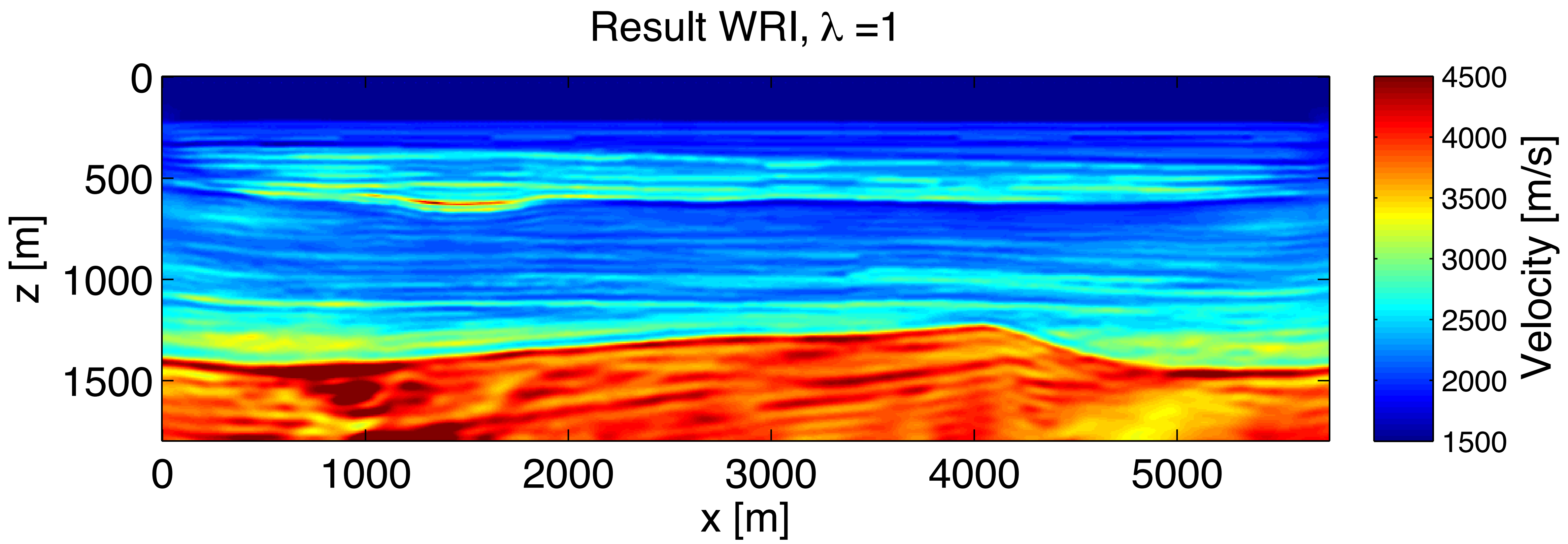


True, initial and final models

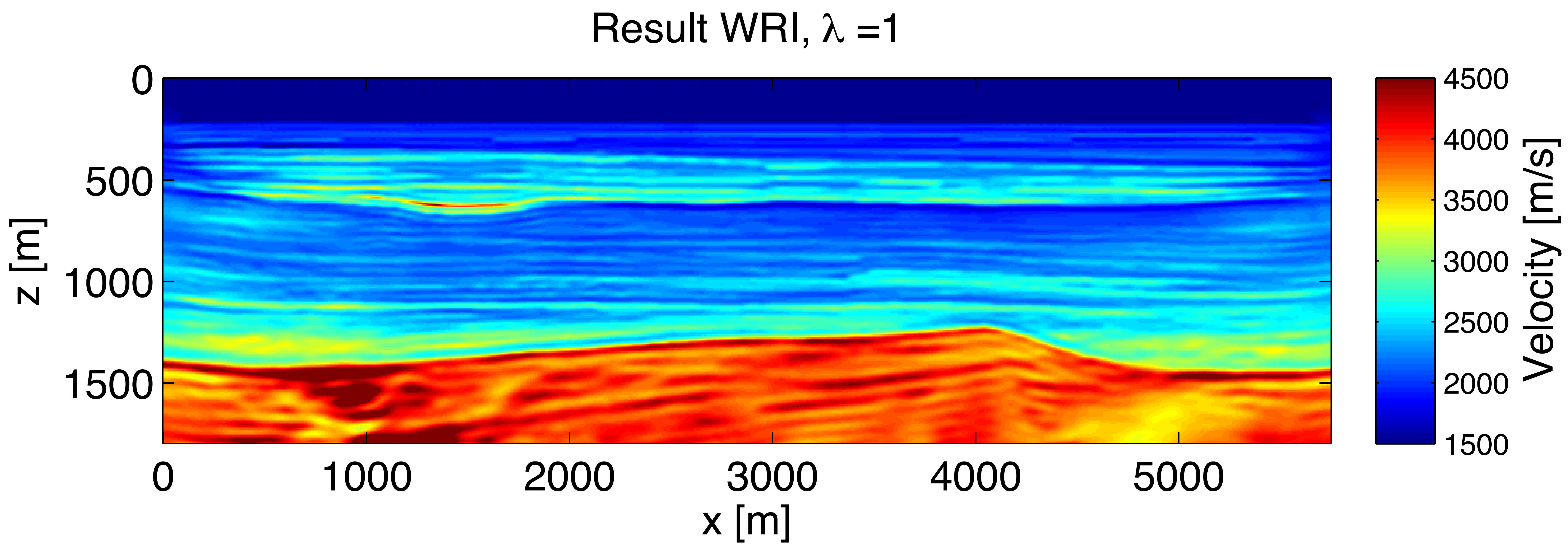




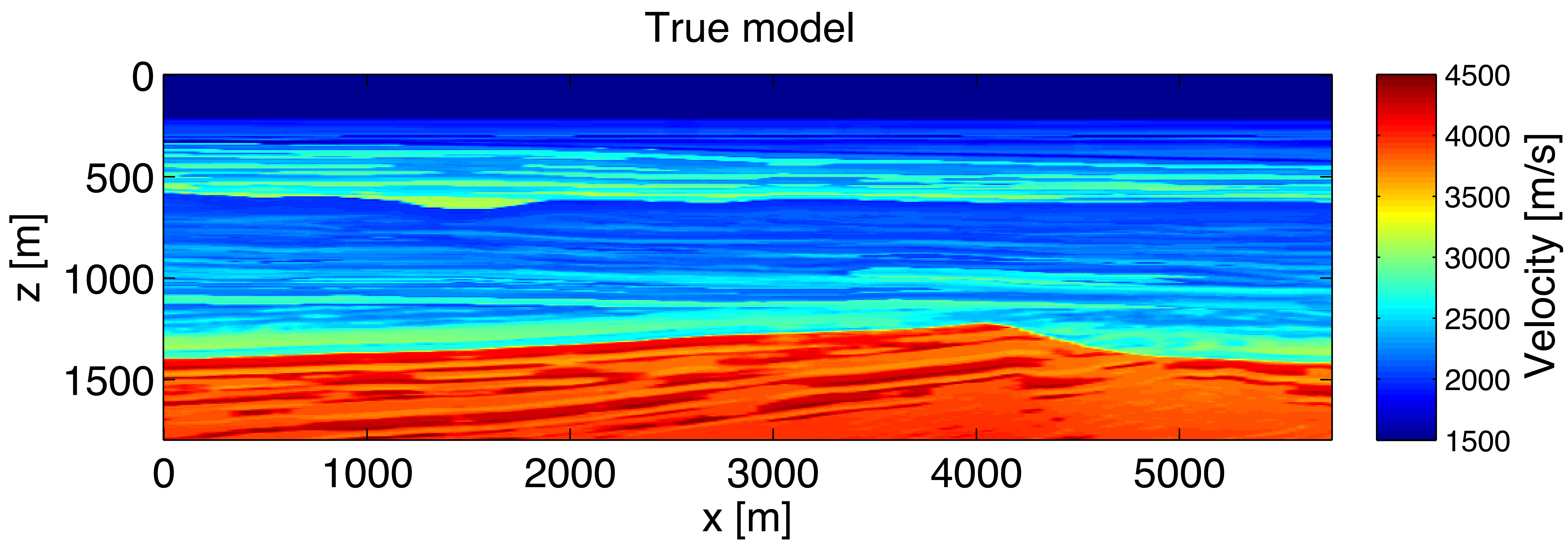
After 2nd cycle



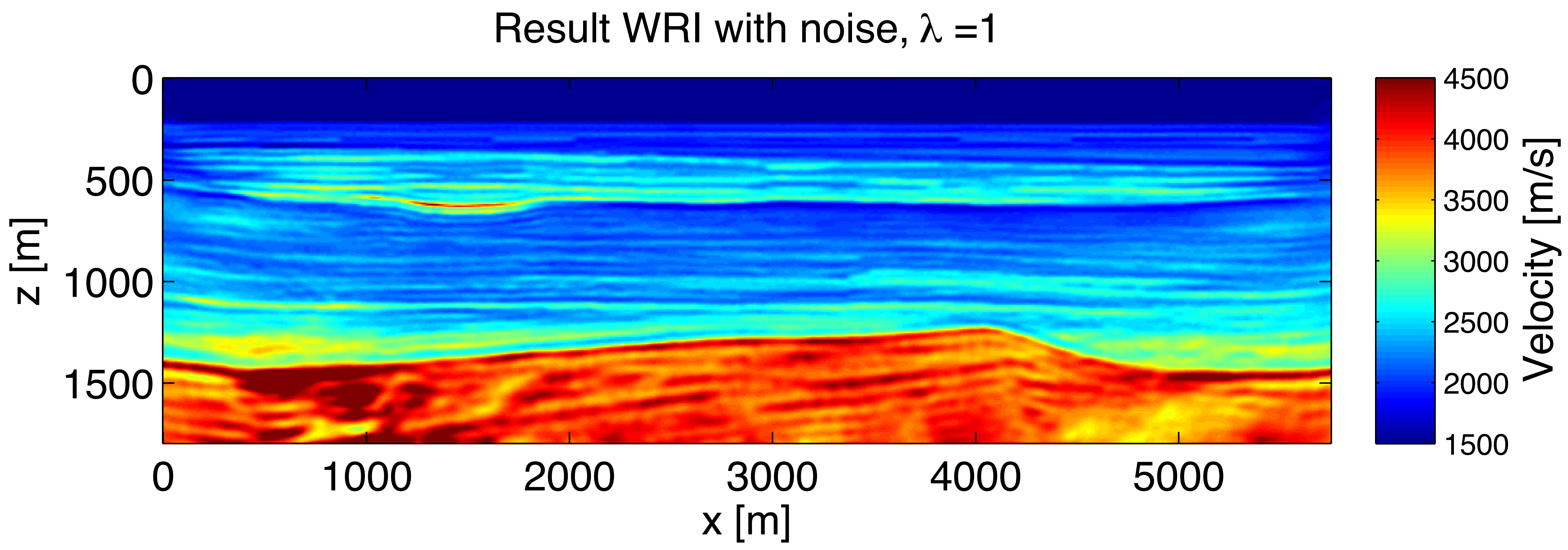
True and final models



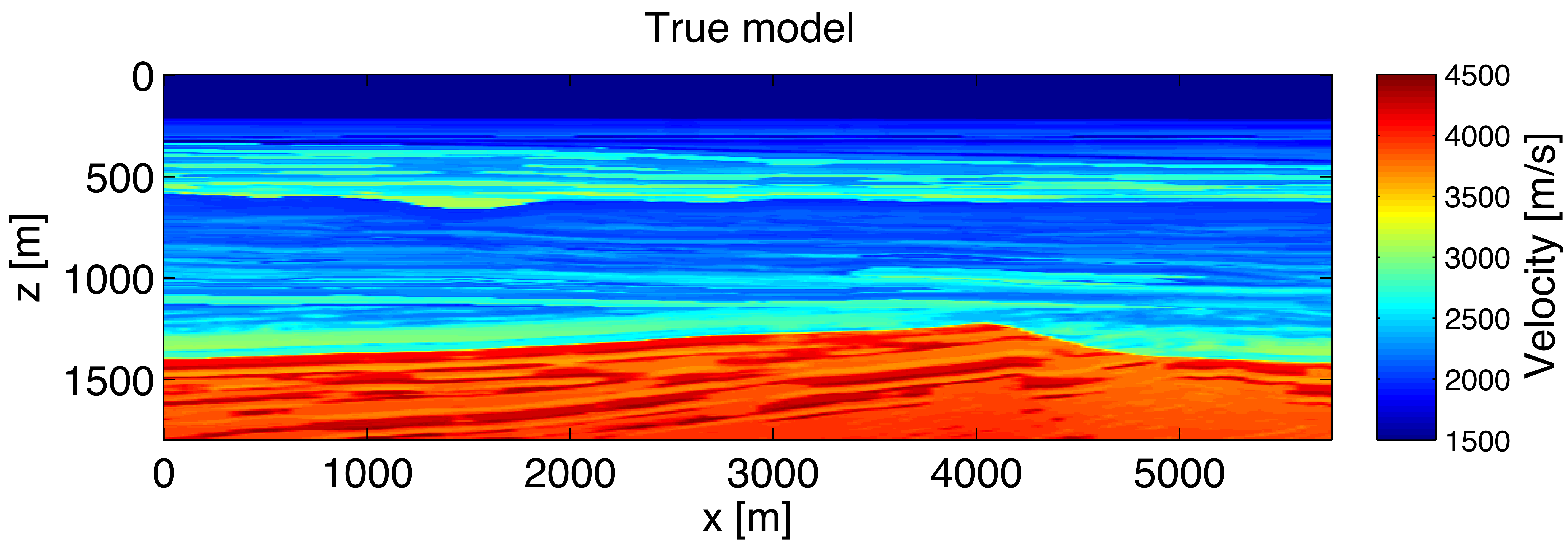
No noise



True and final models



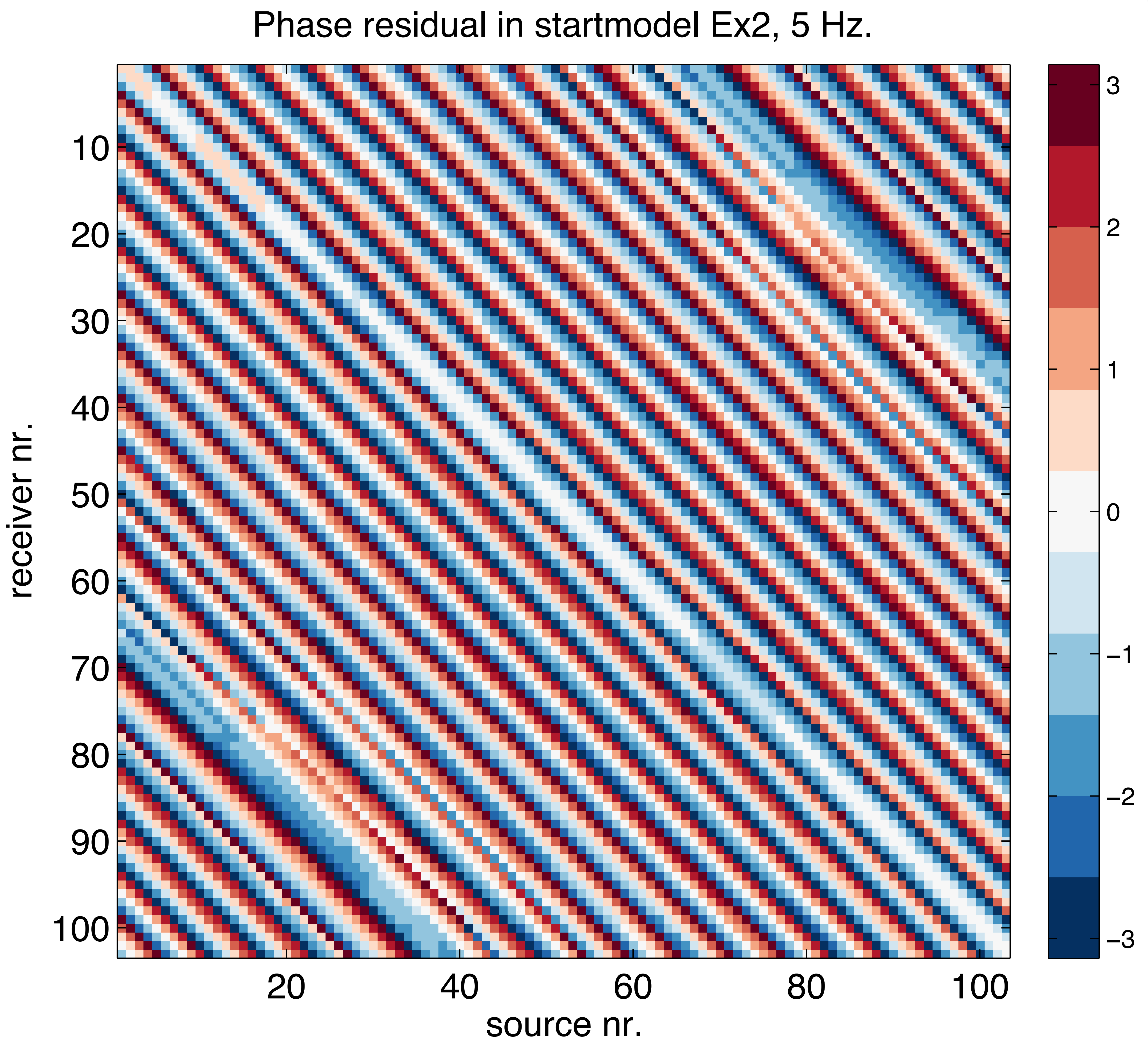
$$\frac{\|\text{noise}\|}{\|\text{data}\|} = 0.04$$



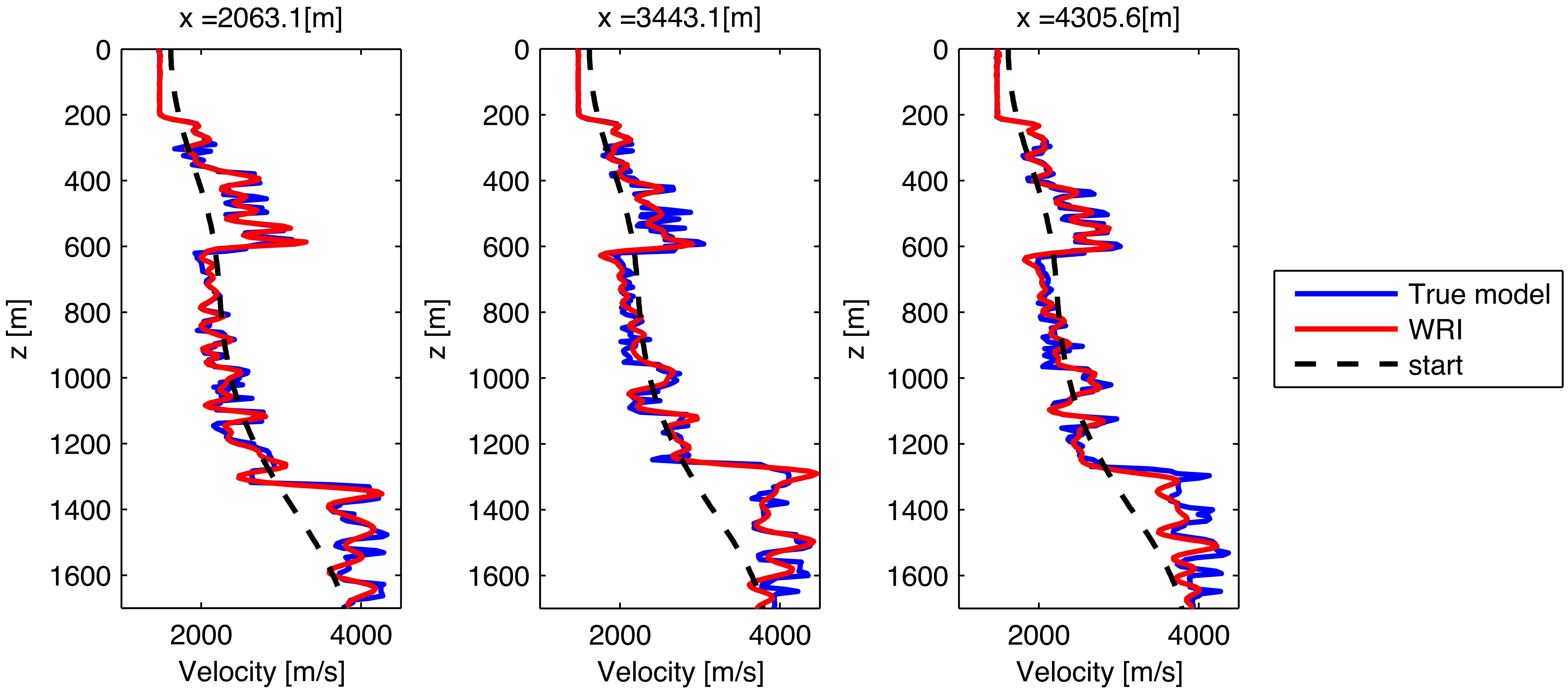
Initial phase-residuals

- Phase residuals computed using the Helmholtz equation in the start model
- WRI does not work with exact wavefields
- WRI uses the 'data-augmented' wavefield
- for λ very small, the phase residual will be 0.
- This will fit noise and cause ill-conditioning (not recommended!)
- Choose λ somewhat balanced (not difficult)

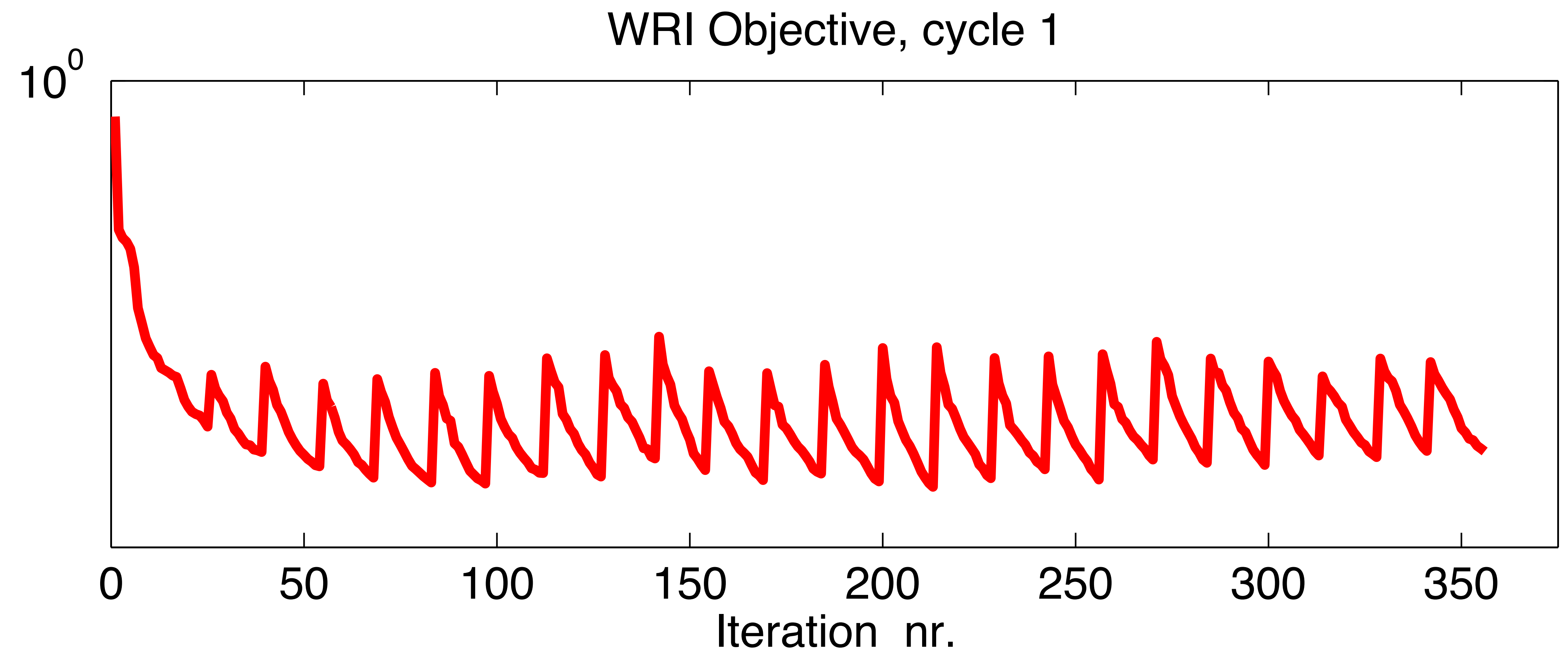
$$\bar{\mathbf{u}}_{kl} = \arg\min_{\mathbf{u}_{kl}} \left\| \begin{pmatrix} \lambda H_k(\mathbf{m}) \\ P \end{pmatrix} \mathbf{u}_{kl} - \begin{pmatrix} \lambda \mathbf{q}_{kl} \\ \mathbf{d}_{kl} \end{pmatrix} \right\|_2$$

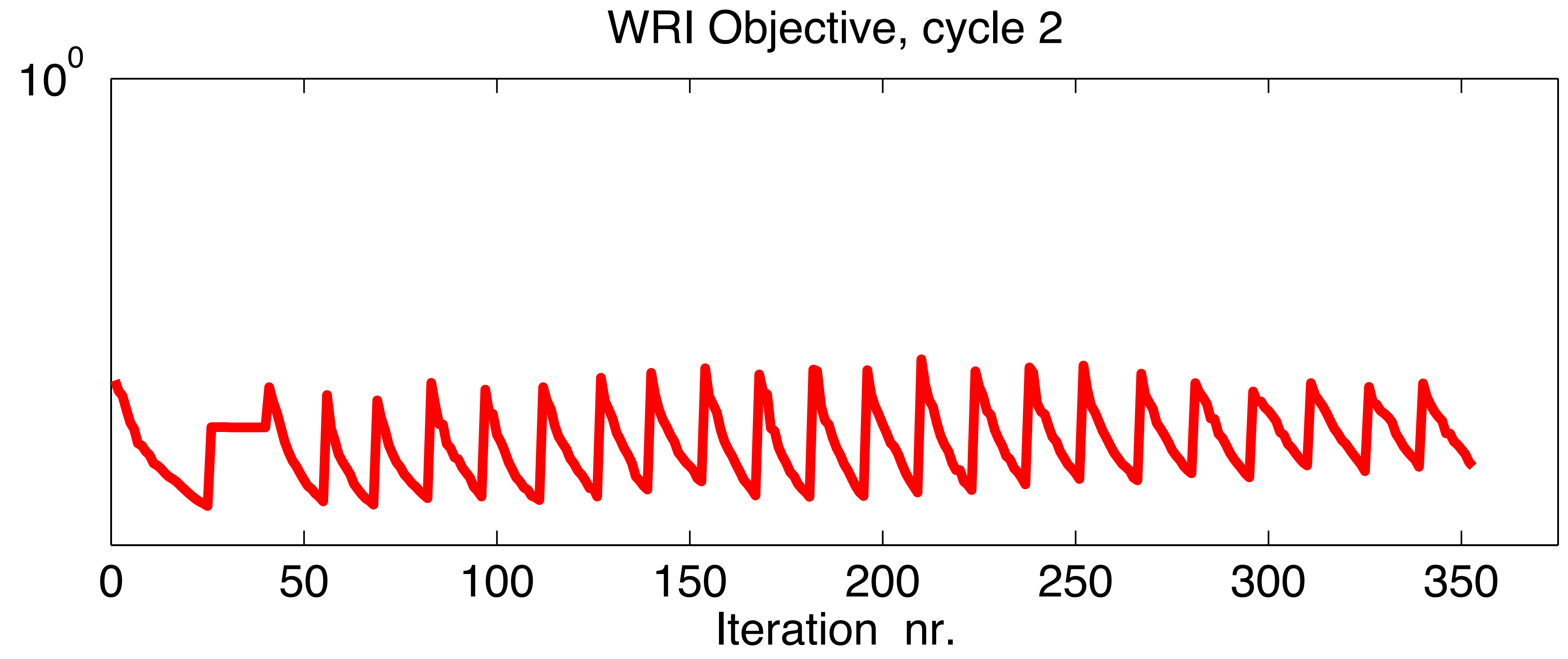


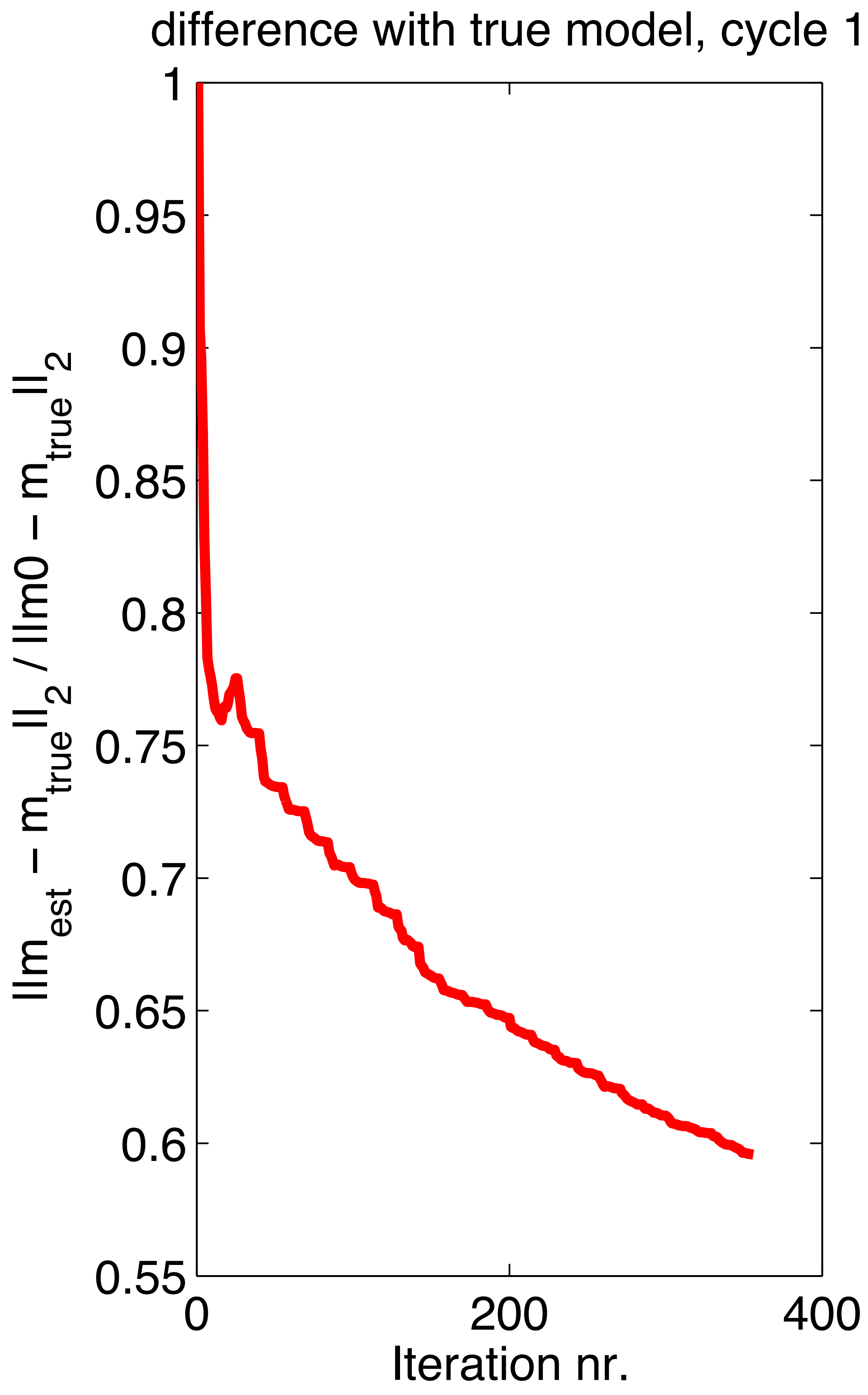
Cross sections

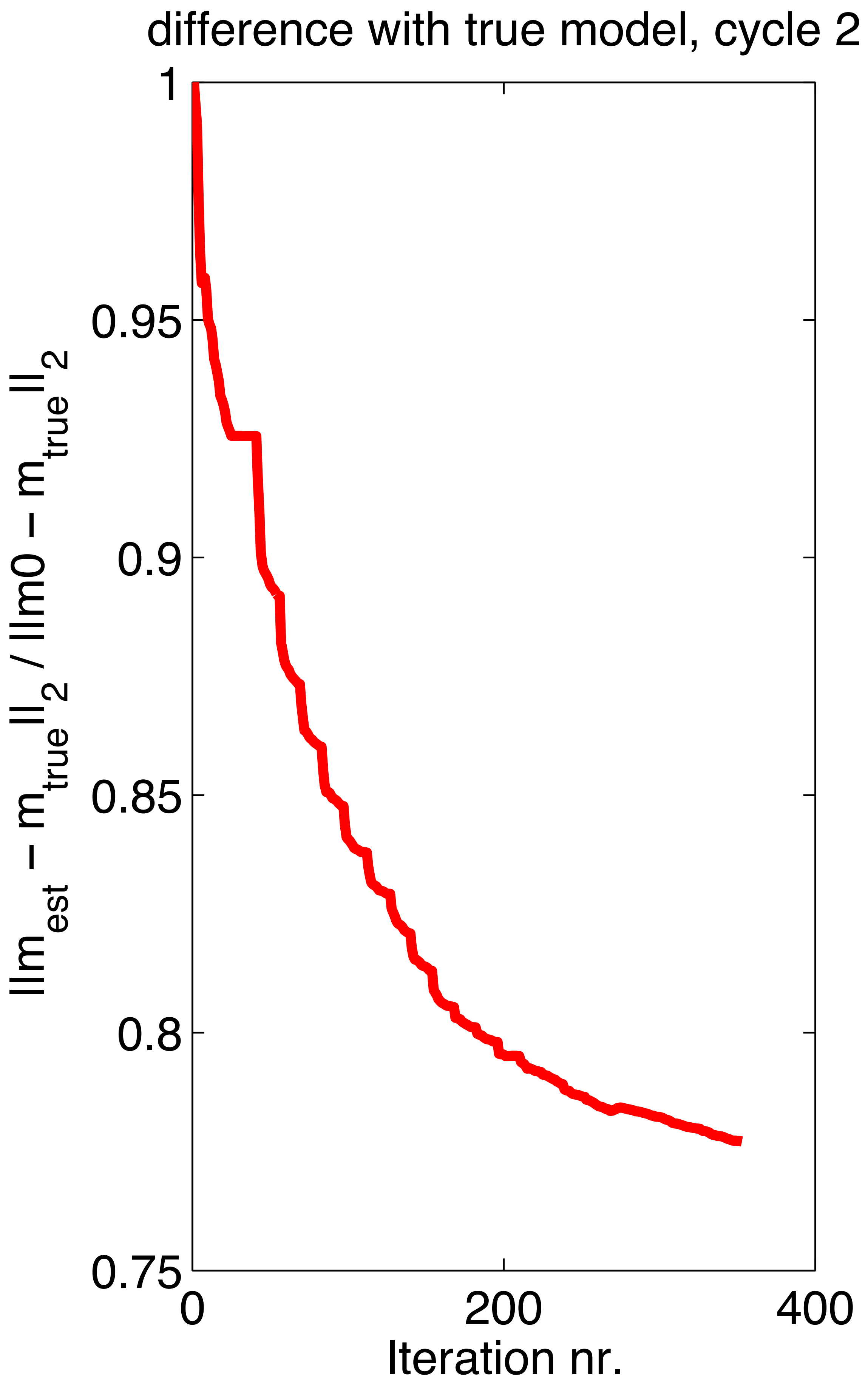


Objective and model error









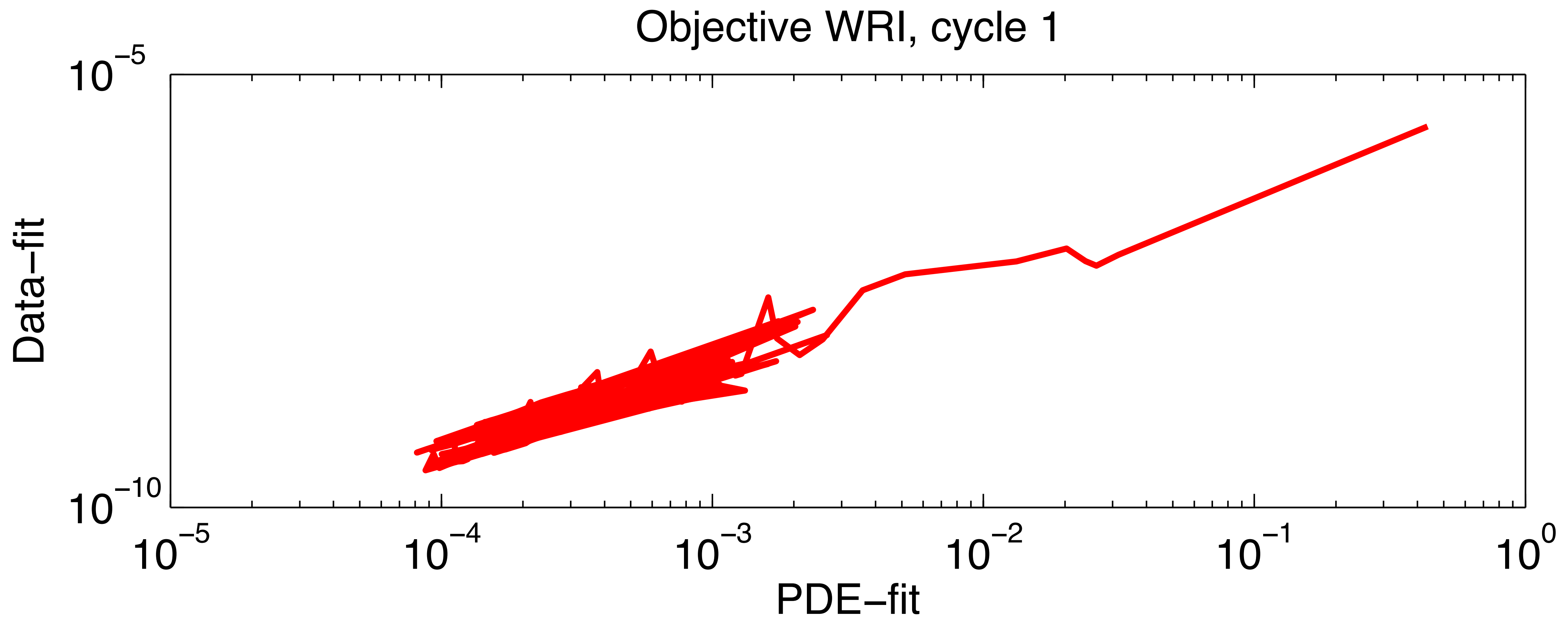
Objective

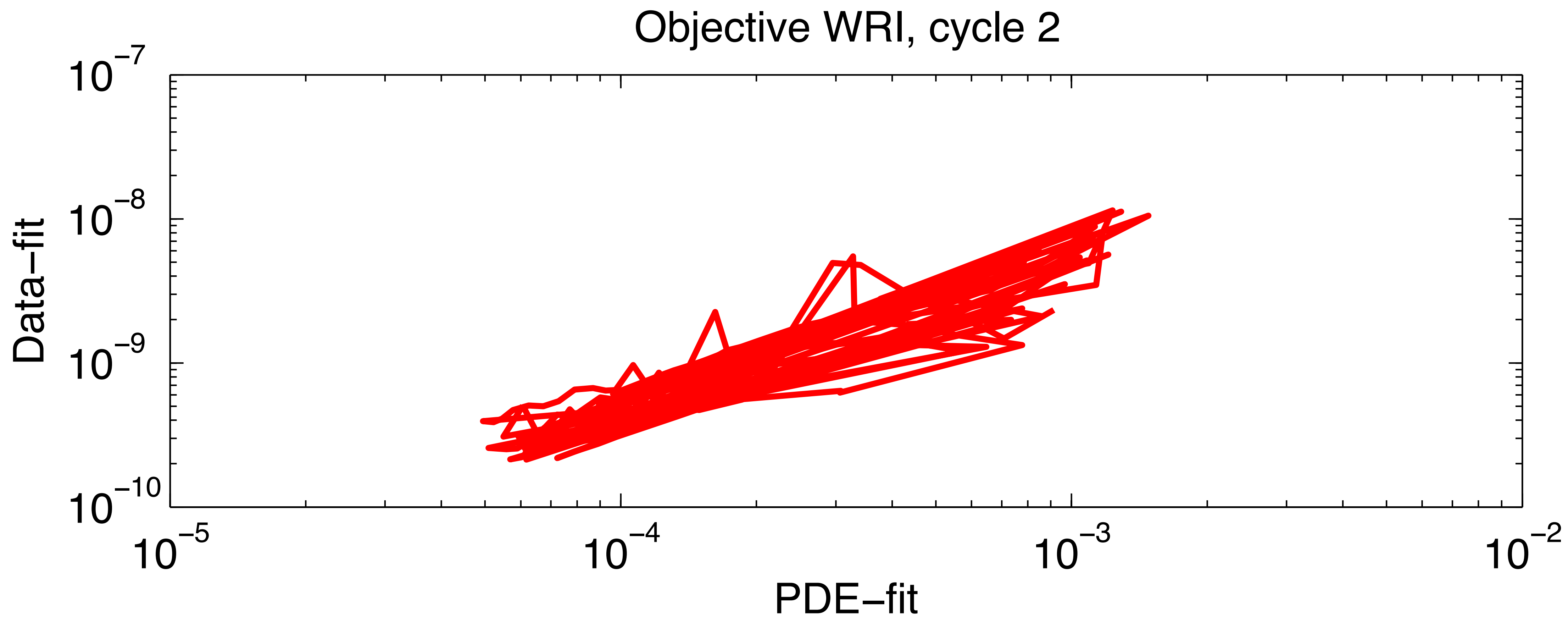
Data-misfit

PDE-misfit

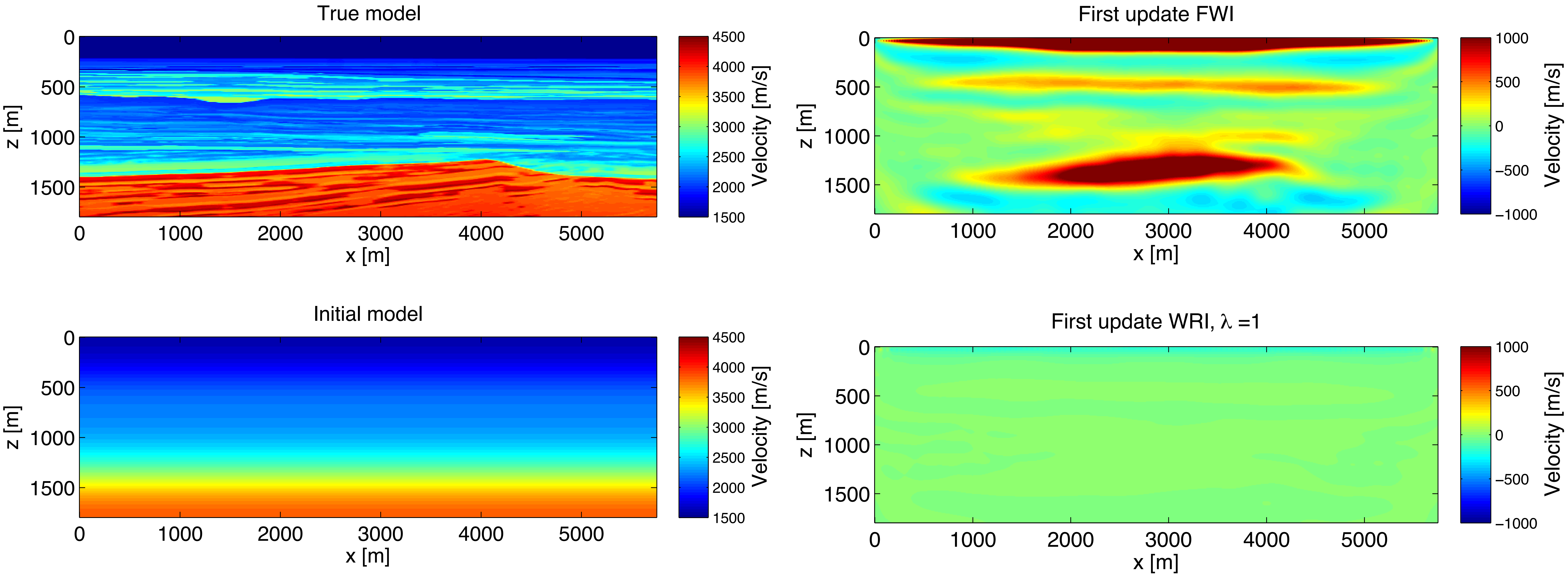
$$\bar{\phi}_\lambda(\mathbf{m}) = \frac{1}{2}\sum_{kl} \|P\bar{\mathbf{u}}_{kl} - \mathbf{d}_{kl}\|_2^2 + \frac{\lambda^2}{2}\|H_k(\mathbf{m})\bar{\mathbf{u}}_{kl} - \mathbf{q}_{kl}\|_2^2$$

- We can take a look at each part separately:
- Data-misfit can go up while iterating!

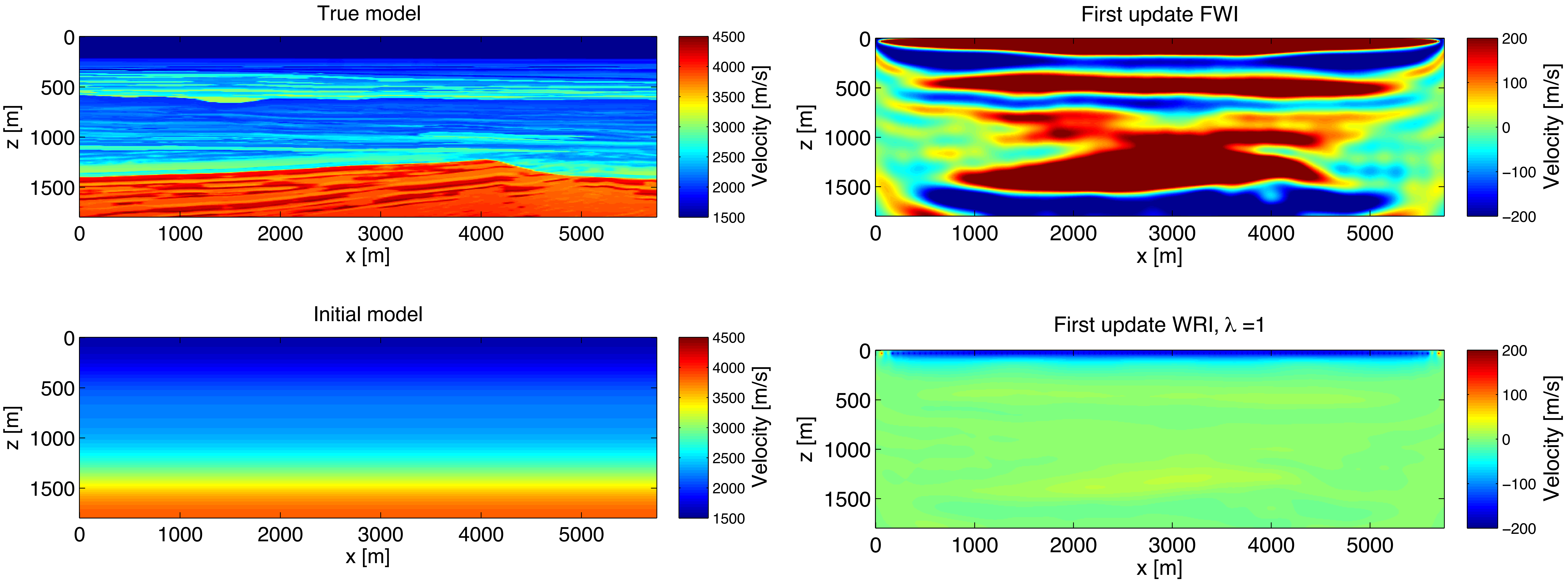




A look at the first update...



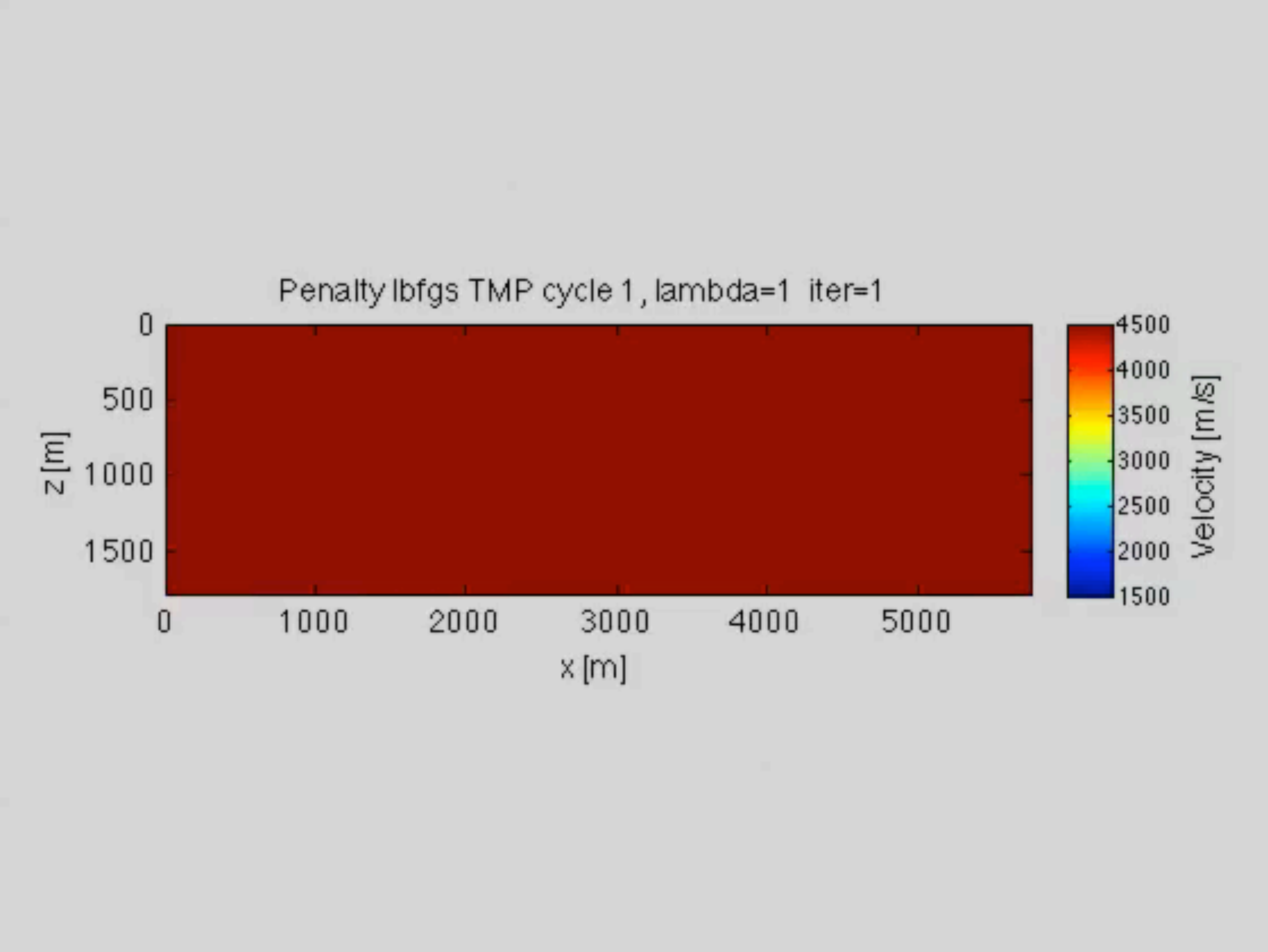
A look at the first update...



Different color scale

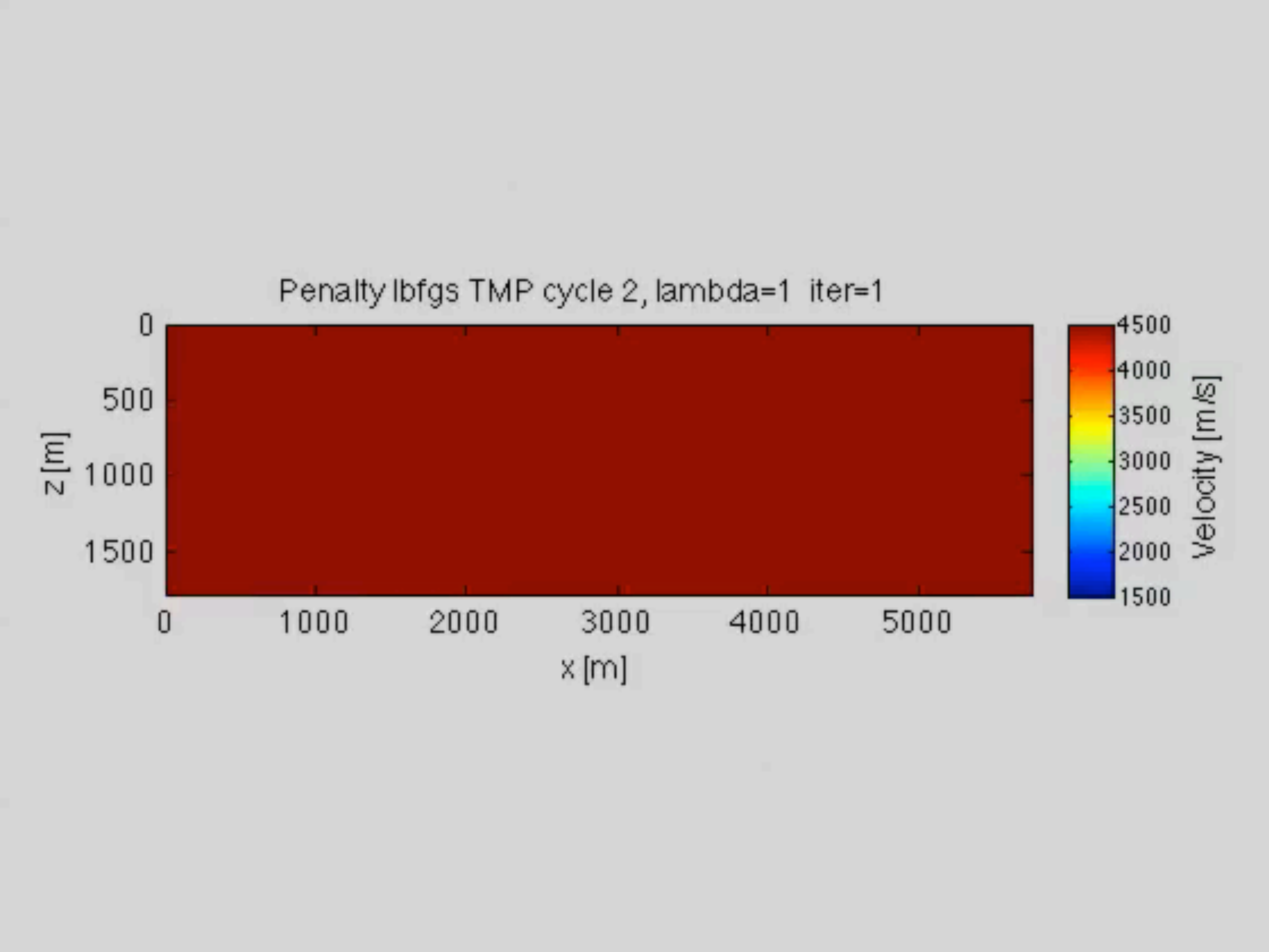
Model evolution

Cycle 1



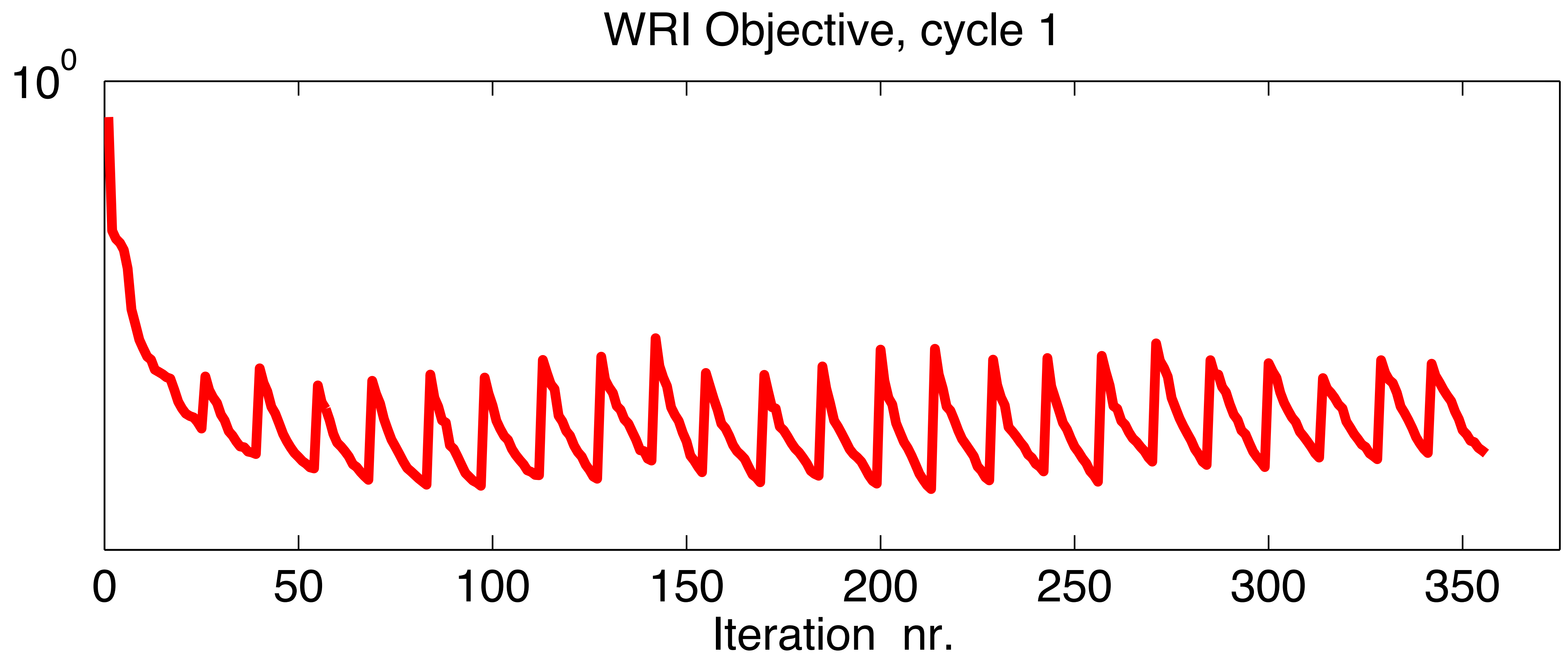
Model evolution

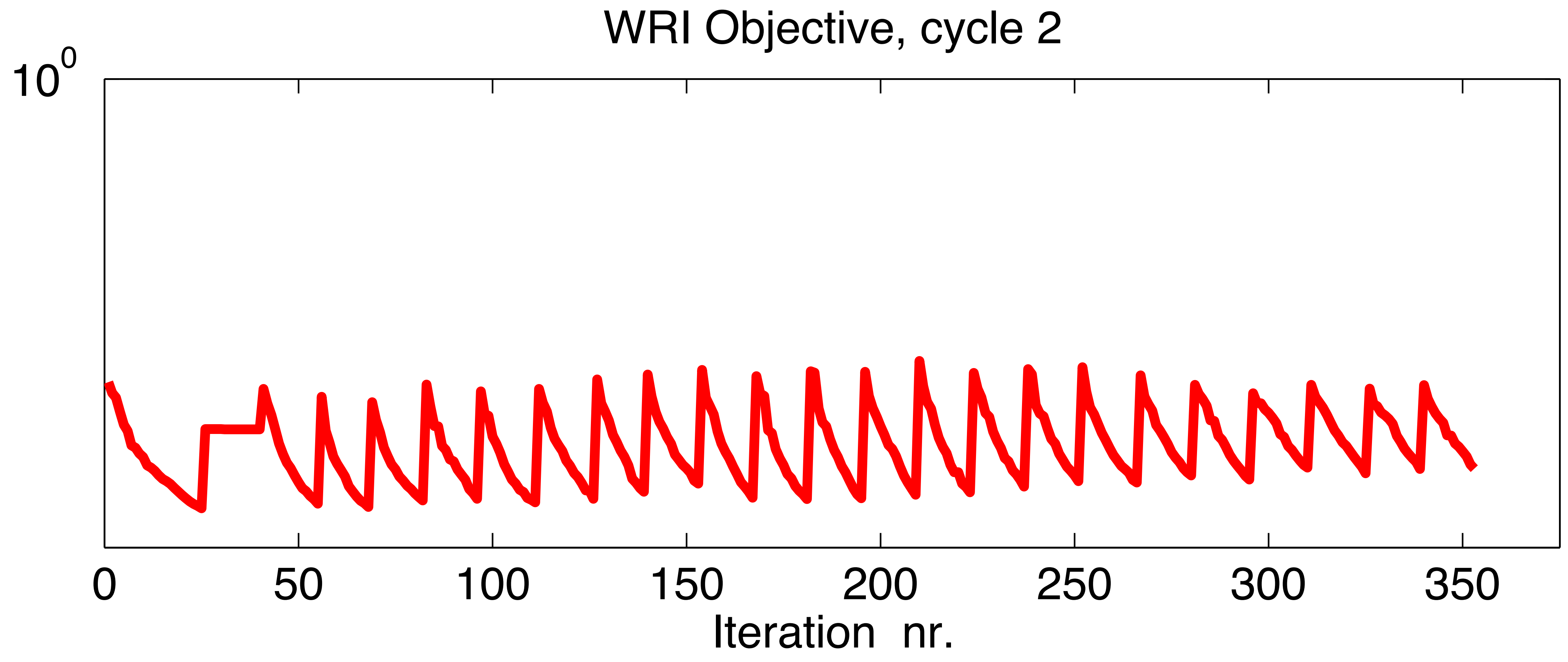
Cycle 2



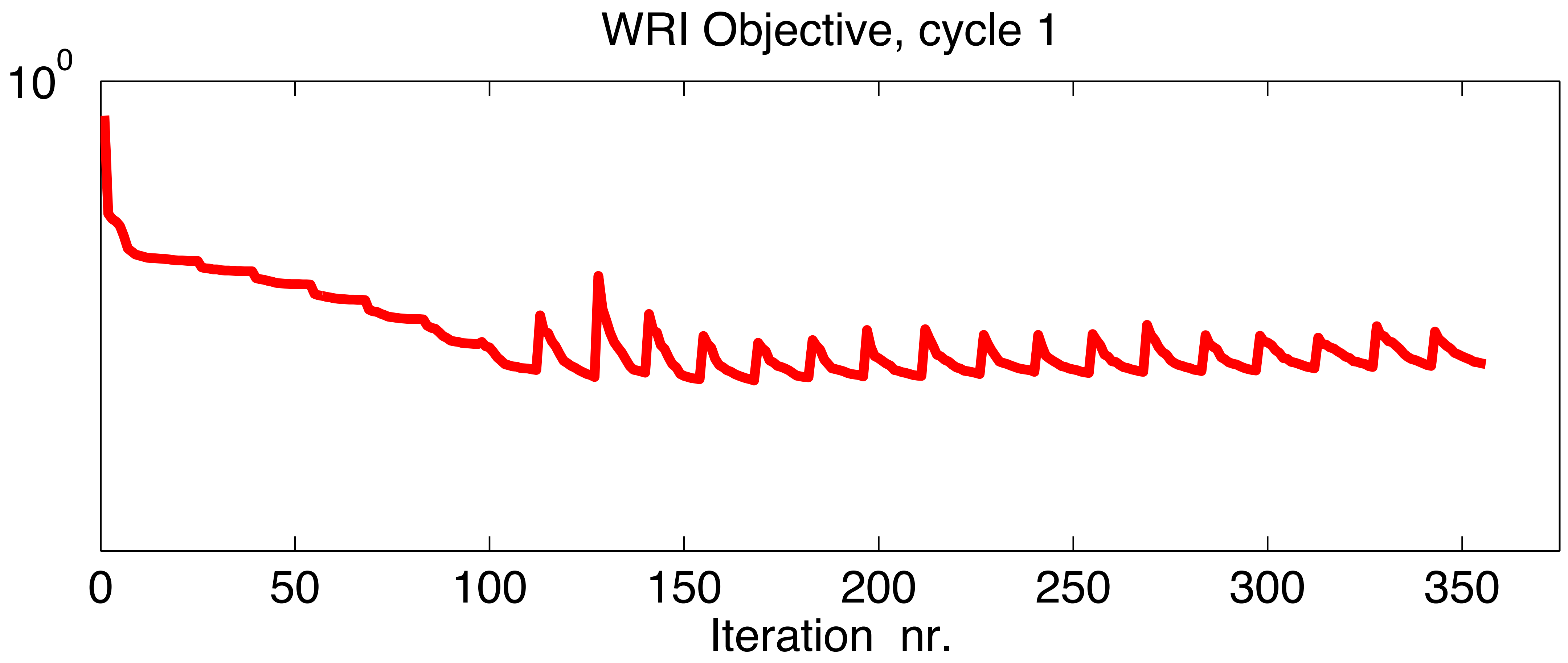
Effect of noise

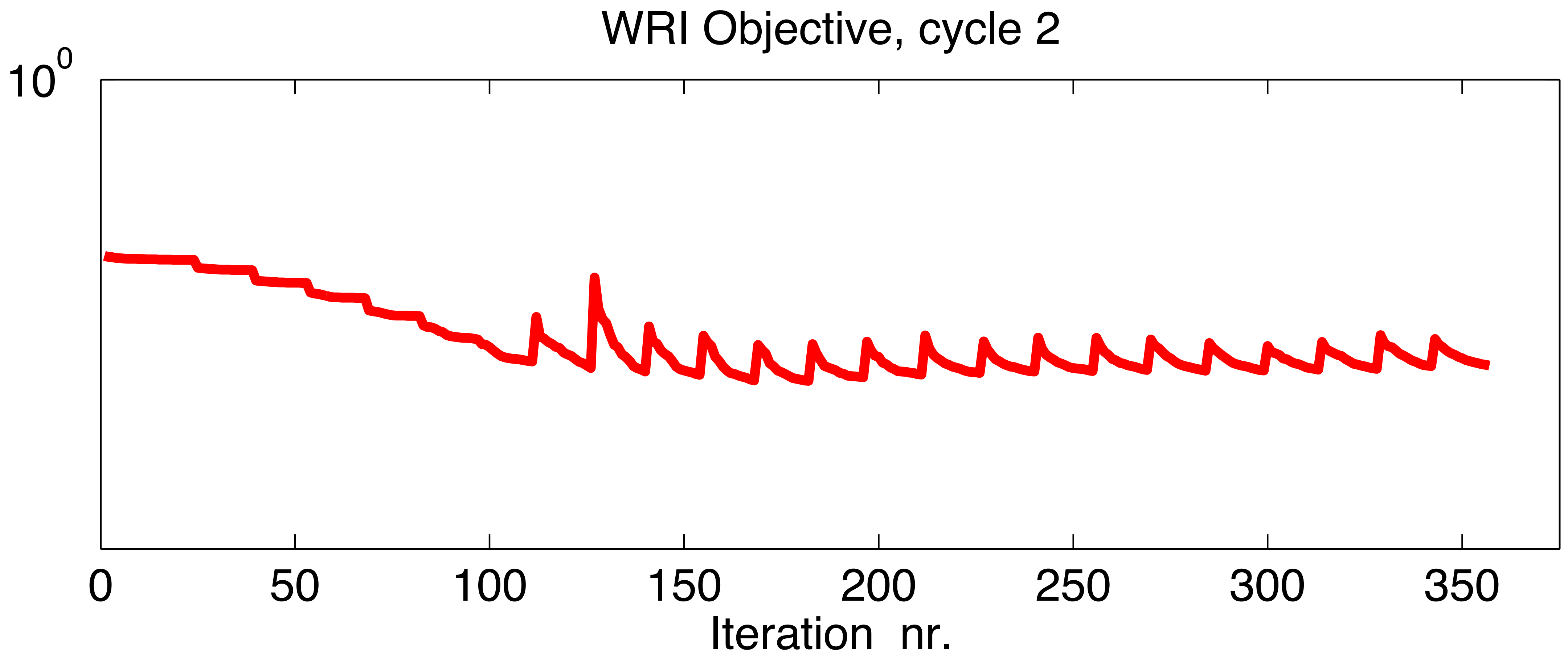
Noise free





Noise





Observations about waveform inversion

- WRI performs much better for difficult problems
- WRI performs similar to FWI for not so difficult problems.

- Even for more difficult problems, only frequency continuation is required.
- No penalty parameter continuation was used, which can potentially increase quality and decrease the number of iterations.

Multi-parameter PDE-constrained optimization

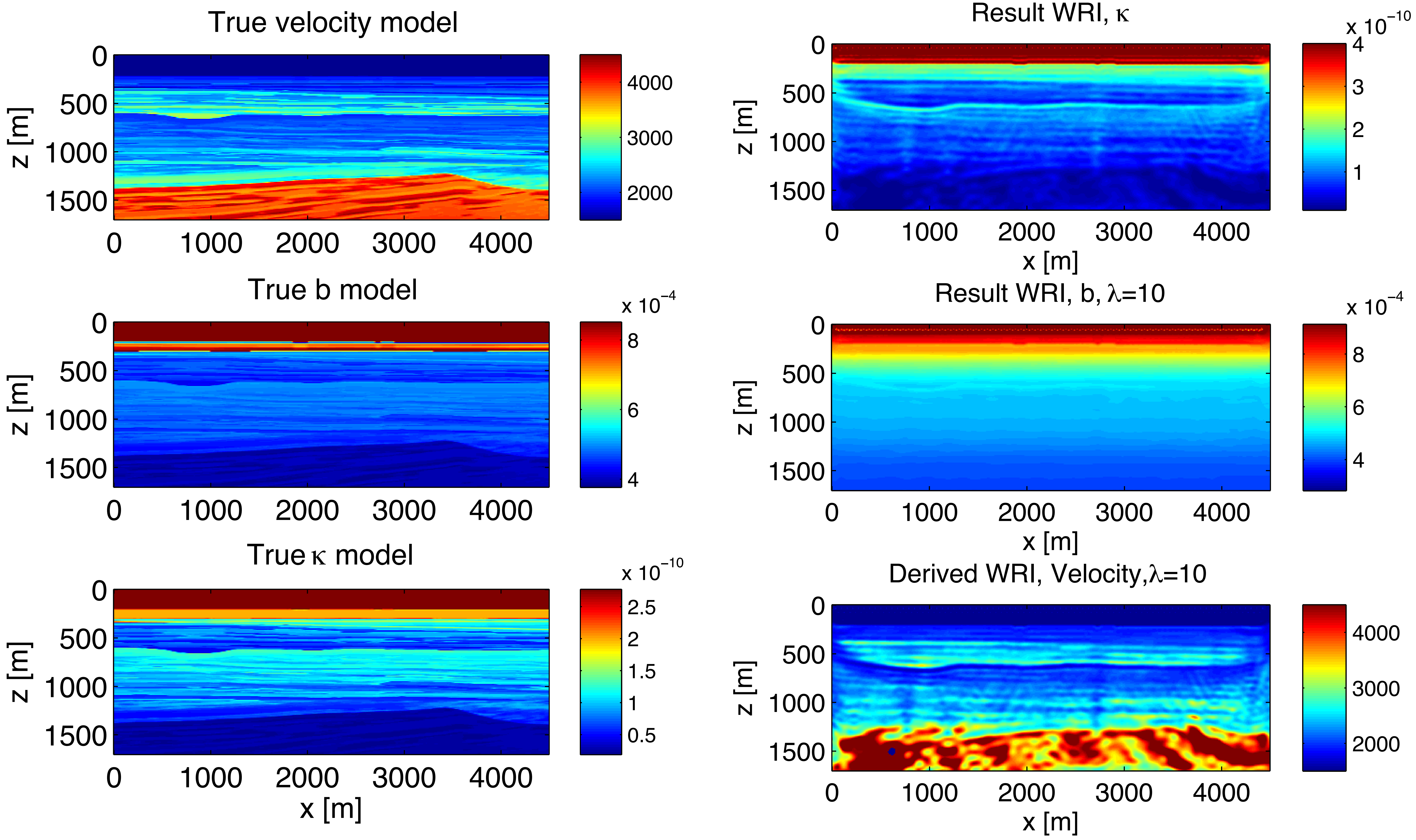
$$\partial_k p + i\omega\rho v_k = f_k$$
$$\partial_r v_r + i\omega\kappa p = q$$

- Difficulty 1: the compressibility and density occur in a coupled system of equations (or in 1 equation when the 2nd order form of the PDE is used)
- Difficulty 2: Different parameters can have very different magnitudes
- => Simple examples show >95.5% of the data can be fitted using only one of the two parameters.

Multi-parameter PDE-constrained optimization

Example: compute gradient of objective and naively plug into L-BFGS

Absolutely unsatisfactory result



SLIM

Multi-parameter PDE-constrained optimization

- Multi-parameter problems of this sort have been solved with little generality
- Proposed solutions include:
    - sequential separate inversion
    - Structural similarity through regularization
    - Manual scaling of gradients corresponding to different parameters
    - Mitigate problems to limited extend by finding the 'best'? parameterization
- All of these require fine-tuning and adaptation to different scenarios
- All of these were designed for gradient-based and quasi-Newton methods

Multi-parameter PDE-constrained optimization

- My approach: Use the actual Hessian
- The Hessian naturally provides information about 'scaling' and 'coupling'.
- Multi-parameter problem:

$$\min_{\mathbf{b},\boldsymbol{\kappa},\mathbf{u}} \frac{1}{2}\|P\mathbf{u}-\mathbf{d}\|_2^2 \quad \text{s.t.} \quad H(\mathbf{b},\boldsymbol{\kappa})\mathbf{u}=\mathbf{q}$$

- Penalty formulation:

$$\phi_\lambda(\mathbf{b},\boldsymbol{\kappa},\mathbf{u}) = \frac{1}{2}\|P\mathbf{u}-\mathbf{d}\|_2^2 + \frac{\lambda^2}{2}\|H(\mathbf{b},\boldsymbol{\kappa})\mathbf{u}-\mathbf{q}\|_2^2$$

[T. van Leeuwen & F.J. Herrmann, 2013]

- Turns out to have additional benefits in this case

A reduced-space algorithm

$$\phi_\lambda(\mathbf{b},\boldsymbol{\kappa},\mathbf{u}) = \frac{1}{2}\|P\mathbf{u}-\mathbf{d}\|_2^2 + \frac{\lambda^2}{2}\|H(\mathbf{b},\boldsymbol{\kappa})\mathbf{u}-\mathbf{q}\|_2^2$$

- Corresponding Newton-system (sparse)

$$\left(\begin{array}{c|cc} \nabla^2_{\mathbf{u},\mathbf{u}}\phi_\lambda & \nabla^2_{\mathbf{u},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\mathbf{u},\mathbf{b}}\phi_\lambda \\ \hline \nabla^2_{\boldsymbol{\kappa},\mathbf{u}}\phi_\lambda & \nabla^2_{\boldsymbol{\kappa},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\boldsymbol{\kappa},\mathbf{b}}\phi_\lambda \\ \nabla^2_{\mathbf{b},\mathbf{u}}\phi_\lambda & \nabla^2_{\mathbf{b},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\mathbf{b},\mathbf{b}}\phi_\lambda \end{array}\right) \begin{pmatrix} \delta_{\mathbf{u}} \\ \delta_{\boldsymbol{\kappa}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{u}}\phi_\lambda \\ \nabla_{\boldsymbol{\kappa}}\phi_\lambda \\ \nabla_{\mathbf{b}}\phi_\lambda \end{pmatrix}$$

write: $\begin{pmatrix} E & B \\ C & D \end{pmatrix}$

- When we solve for the fields as before, $\left(\nabla_{\mathbf{u}}\phi_\lambda(\bar{\mathbf{u}},\boldsymbol{\kappa},\mathbf{b}) = 0\right)$ we obtain

$$\underset{\text{sparse}}{(D} - \underset{\text{dense}}{C}E^{-1}B)\begin{pmatrix} \delta_{\boldsymbol{\kappa}} \\ \delta_{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \nabla_{\boldsymbol{\kappa}}\phi_\lambda \\ \nabla_{\mathbf{b}}\phi_\lambda \end{pmatrix} - CE^{-1}\underbrace{\nabla_{\mathbf{u}}\phi_\lambda(\bar{\mathbf{u}},\boldsymbol{\kappa},\mathbf{b})}_{=0}$$

- Sparse-Dense splitting does not happen in the Lagrangian form.

A reduced-space algorithm

$$\big(\overset{\text{sparse}}{D} - \overset{\text{dense}}{C}E^{-1}B\big)\begin{pmatrix}\delta_{\boldsymbol{\kappa}}\\ \delta_{\mathbf{u}}\end{pmatrix} = \begin{pmatrix}\nabla_{\boldsymbol{\kappa}}\phi_\lambda\\ \nabla_{\mathbf{b}}\phi_\lambda\end{pmatrix} - CE^{-1}\overset{=0}{\nabla_{\mathbf{u}}\phi_\lambda(\bar{\mathbf{u}},\boldsymbol{\kappa},\mathbf{b})}$$

$$D = \begin{pmatrix}\nabla^2_{\boldsymbol{\kappa},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\boldsymbol{\kappa},\mathbf{b}}\phi_\lambda\\ \nabla^2_{\mathbf{b},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\mathbf{b},\mathbf{b}}\phi_\lambda\end{pmatrix}$$

We can use this as an approximate Hessian

So far I used only the diagonal blocks, yielding a sparse, SPD matrix

$$\tilde{H} = \begin{pmatrix}\nabla^2_{\boldsymbol{\kappa},\boldsymbol{\kappa}}\phi_\lambda & 0\\ 0 & \nabla^2_{\mathbf{b},\mathbf{b}}\phi_\lambda\end{pmatrix} = \begin{pmatrix}G^*_{\boldsymbol{\kappa}}G_{\boldsymbol{\kappa}} & 0\\ 0 & G^*_{\mathbf{b}}G_{\mathbf{b}}\end{pmatrix}$$

$$G_{\boldsymbol{\kappa}} = \partial H(\mathbf{b},\boldsymbol{\kappa})\bar{\mathbf{u}}/\partial\boldsymbol{\kappa} \qquad G_{\mathbf{b}} = \partial H(\mathbf{b},\boldsymbol{\kappa})\bar{\mathbf{u}}/\partial\mathbf{b}$$

SLIM

A reduced-space algorithm

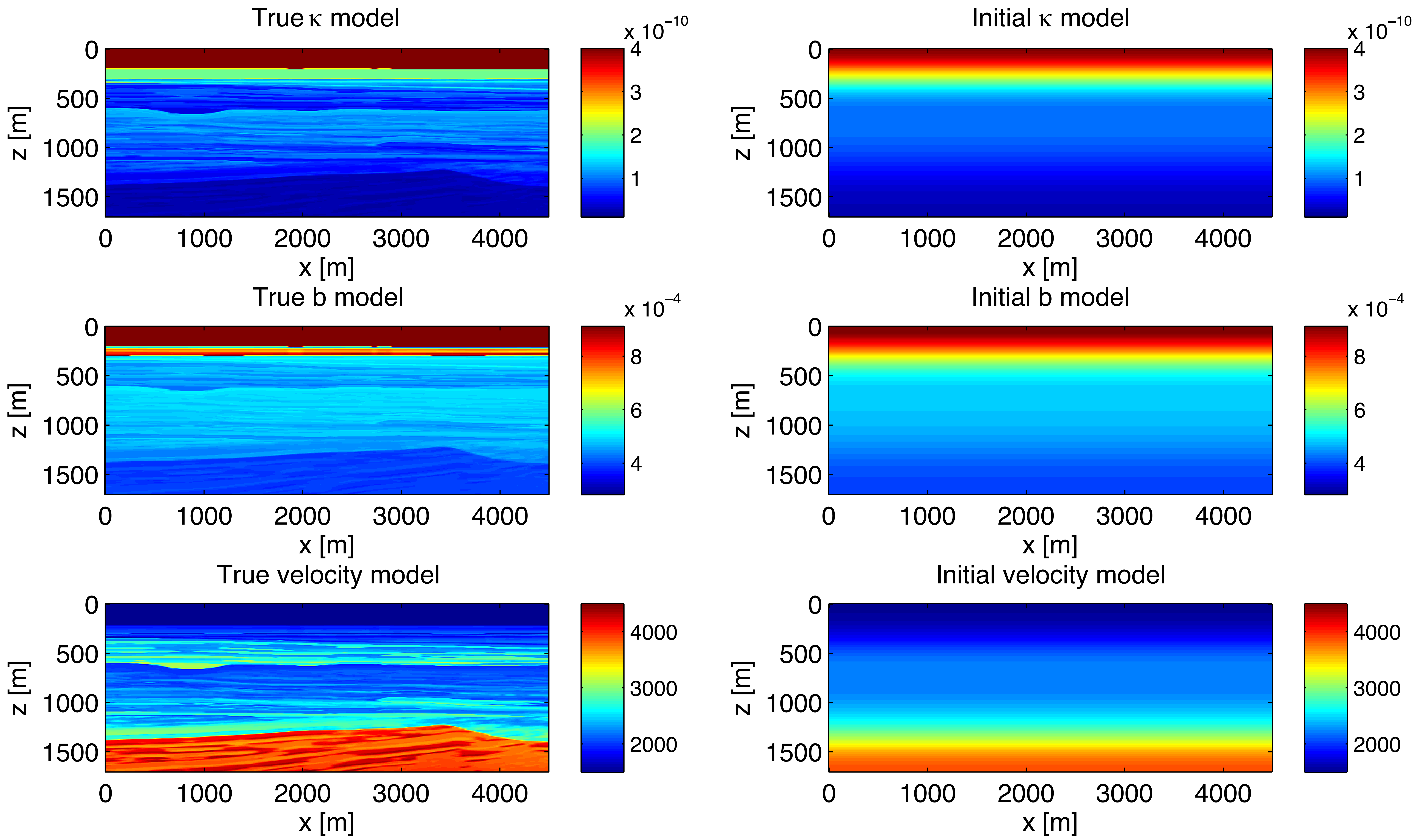
Algorithm 1 Waveform inversion with a sparse Hessian approximation.

while Not converged **do**

1. $\bar{\mathbf{u}} = \arg\min_{\mathbf{u}} \left\| \begin{pmatrix} \lambda \mathbf{A}(\mathbf{b}, \kappa) \\ \mathbf{P} \end{pmatrix} \mathbf{u} - \begin{pmatrix} \lambda \mathbf{q} \\ \mathbf{d} \end{pmatrix} \right\|_2$ `// Solve`
2. $\mathbf{G}_\kappa, \mathbf{G}_\mathbf{b}, \nabla_\mathbf{b} \bar{\phi}_\lambda, \nabla_\kappa \bar{\phi}_\lambda$ `// Form`
3. $\mathbf{p}_{gn} = \tilde{\mathbf{H}}^{-1} \mathbf{g}$ `// Solve`
4. find steplength α `// Linesearch`
5. $\mathbf{m} = \mathbf{m} + \alpha \mathbf{p}_{gn}$ `// update model`

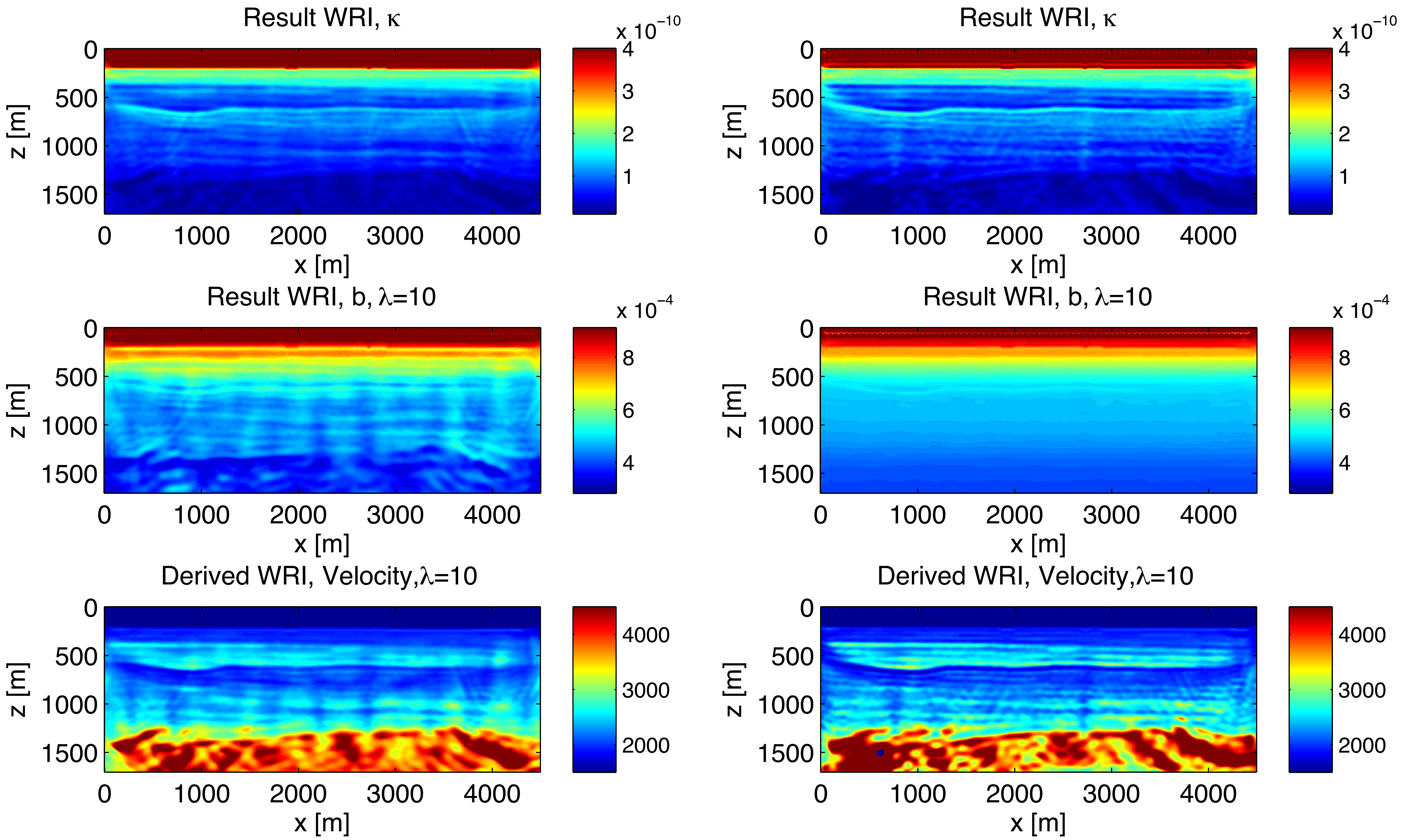
end

A reduced-space algorithm; example



A reduced-space algorithm; example

New algorithm



basic L-BFGS

A reduced-space algorithm

- Spare Hessian approximation seems to be quite effective
- Is it sufficient? Needs more experimenting...
- Alternatively, we can use a matrix-free Newton-CG algorithm including the dense part of the reduced Hessian.

$$\big(\overset{\text{sparse}}{D} - \overset{\text{dense}}{CE^{-1}B}\big)\begin{pmatrix}\delta_{\boldsymbol{\kappa}}\\ \delta_{\mathbf{u}}\end{pmatrix} = \begin{pmatrix}\nabla_{\boldsymbol{\kappa}}\phi_\lambda\\ \nabla_{\mathbf{b}}\phi_\lambda\end{pmatrix} - CE^{-1}\overset{=0}{\nabla_{\mathbf{u}}\phi_\lambda}(\bar{\mathbf{u}}, \boldsymbol{\kappa}, \mathbf{b})$$

Conclusions

- The use of the Hessian automatically scales the different parameter classes based on the PDE itself.
- Hessian information may turn multi-parameter inversion algorithms fully automatic.
- A quadratic-penalty formulation offers very cheap and effective Hessian approximations.
- Promising results using the multi-parameter WRI version.

SLIM

Work in progress

- If you are willing to have more than 2 wavefields in memory, the door too many optimization algorithms is opened.
- For Lagrangian as well as penalty methods.
- Access to sparse Hessians, gradients without solving PDE's.
- Function evaluations do not require PDE solves either .
- Update the fields instead of solving.

$$\phi_\lambda(\mathbf{b}, \boldsymbol{\kappa}, \mathbf{u}) = \frac{1}{2}\|P\mathbf{u} - \mathbf{d}\|_2^2 + \frac{\lambda^2}{2}\|H(\mathbf{b}, \boldsymbol{\kappa})\mathbf{u} - \mathbf{q}\|_2^2$$

Newton step ➔ $$\left(\begin{array}{c|cc} \nabla^2_{\mathbf{u},\mathbf{u}}\phi_\lambda & \nabla^2_{\mathbf{u},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\mathbf{u},\mathbf{b}}\phi_\lambda \\ \hline \nabla^2_{\boldsymbol{\kappa},\mathbf{u}}\phi_\lambda & \nabla^2_{\boldsymbol{\kappa},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\boldsymbol{\kappa},\mathbf{b}}\phi_\lambda \\ \nabla^2_{\mathbf{b},\mathbf{u}}\phi_\lambda & \nabla^2_{\mathbf{b},\boldsymbol{\kappa}}\phi_\lambda & \nabla^2_{\mathbf{b},\mathbf{b}}\phi_\lambda \end{array}\right) \begin{pmatrix} \delta_{\mathbf{u}} \\ \delta_{\boldsymbol{\kappa}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{u}}\phi_\lambda \\ \nabla_{\boldsymbol{\kappa}}\phi_\lambda \\ \nabla_{\mathbf{b}}\phi_\lambda \end{pmatrix}$$

SLIM

Acknowledgements

The SLIM students & postdocs

SINBAD

NSERC CRSNG

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, CGG, Chevron, ConocoPhillips, ION, Petrobras, PGS, Total SA, WesternGeco, and Woodside.