- Gabrio Rizzuti, Mathias Louboutin, Rongrong Wang, and Felix J. Herrmann, “**A dual formulation of wavefield reconstruction inversion for large-scale seismic inversion**”. 2020.BibTeX

Time-domain Wavefield Reconstruction Inversion in Transvers Tilted Isotropic media.

M. Louboutin*, G. Rizzuti, R. Wang and F. J. Herrmann

SLIM
Georgia Institute of Technology

Motivations

Non-convex, non-linear, large scale PDE constraints optimization

Time-harmonic solutions:

- Implicit solvers (damped/extendend lsqr, normal equations, ...)
- Not scalable (or hard to)
- Limited physical representations

Time domain:

- Scalable
- Explicit solver (time marching only)

PDE-constrained optimization

$$\min_{\mathbf{m},\mathbf{u}} \frac{1}{2}||\mathbf{d} - R\,\mathbf{u}||^2 \quad \text{subject to} \quad A(\mathbf{m})\,\mathbf{u} = \mathbf{q}$$

Vectors:

- $\mathbf{d}$ data
- $\mathbf{q}$ source
- $\mathbf{m}$ model
- $\mathbf{u}$ wavefield

Operators:

- $A(\mathbf{m})$ wave equation
- R receiver restriction

PDE-constrained optimization (all-at-once full-space)

$$\max_{\mathbf{v}} \min_{\mathbf{m},\mathbf{u}} \mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{v}), \quad \mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{v}) = \frac{1}{2}||\mathbf{d} - R\,\mathbf{u}||^2 + \langle \mathbf{v}, \mathbf{q} - A(\mathbf{m})\,\mathbf{u} \rangle$$

[Haber, E., and Ascher, U. M., 2001; Biros, G., and Ghattas, O., 2005; Grote et al, 2011]

Pros:

✓ **cheap** evaluation and gradient computation (no need for PDE solution)

Cons:

✗ **need simultaneous storage of wavefields and multipliers (for each source and time/frequency sample)**

PDE-constrained optimization (reduced space)

$$\min_{\mathbf{m}} J(\mathbf{m}), \quad J(\mathbf{m}) = \frac{1}{2}||\mathbf{d} - F(\mathbf{m})\,\mathbf{q}||^2, \quad F(\mathbf{m}) = R\,A(\mathbf{m})^{-1}$$

[Tarantola, A., '84; Haber, E., et al, 2000; Epanomeritakis, I., et al, 2008]

Pros:

✓ no simultaneous **wavefield storage** for all sources and frequencies (compute and discard)

Cons:

✗ highly **non-convex** (needs good starting model)

✗ requires exact **PDE solutions** (prohibitive in 3D for frequency domain)

PDE-constrained optimization (penalty method)

$$J_\lambda(\mathbf{m},\mathbf{u}) = \frac{1}{2}||\mathbf{d} - R\,\mathbf{u}||^2 + \frac{\lambda^2}{2}\underbrace{||\mathbf{q} - A(\mathbf{m})\,\mathbf{u}||^2}_{\text{PDE penalty}}$$

[van den Berg, P. M., and Kleinman, R. E., 1997; van Leeuwen, T. and Herrmann, F. J., 2013]

PDE-constrained optimization (WRI)

$$J_\lambda(\mathbf{m}) = \frac{1}{2}||\mathbf{d} - R\,\bar{\mathbf{u}}||^2 + \frac{\lambda^2}{2}||\mathbf{f} - A(\mathbf{m})\,\bar{\mathbf{u}}||^2$$

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

$$\begin{bmatrix} R \\ \lambda A(\mathbf{m}) \end{bmatrix} \bar{\mathbf{u}} = \begin{bmatrix} \mathbf{d} \\ \lambda \mathbf{q} \end{bmatrix}$$

augmented wave equation

Pros:

✓ no simultaneous **wavefield storage** for all sources and frequencies (compute and discard)

Cons:

✗ needs **augmented PDE solution**

✗ frequency domain: does not effectively scale to 3D

✗ time domain: no explicit time-marching scheme

Early attempt to circumvent WRI shortcomings

$$J_\lambda(\mathbf{m}) = \frac{1}{2}||\mathbf{d} - F(\mathbf{m})\,\bar{\mathbf{q}}||^2 + \frac{\lambda^2}{2}||\mathbf{q} - \bar{\mathbf{q}}||^2$$

[Wang et al, 2016, Huang et al, 2018]:

$$\begin{bmatrix} F(\mathbf{m}) \\ \lambda I \end{bmatrix} \bar{\mathbf{q}} = \begin{bmatrix} \mathbf{d} \\ \lambda \mathbf{q} \end{bmatrix}$$

augmented wave equation

Variable projection **approximation**:

$$\bar{\mathbf{q}} \approx \mathbf{q} + \frac{1}{\lambda^2} F(\mathbf{m})^* (\mathbf{d} - F(\mathbf{m})\,\mathbf{q}), \quad \lambda \to \infty$$

Denoising reformulation of WRI

$$\min_{\mathbf{m},\mathbf{u}} \frac{1}{2} \underbrace{||\mathbf{q} - A(\mathbf{m})\,\mathbf{u}||^2}_{\text{PDE misfit}} \quad \text{s.t.} \quad \underbrace{||\mathbf{d} - R\,\mathbf{u}|| \leq \varepsilon}_{\text{data constraint}}$$

[Wang, R., and Herrmann, F. J., 2017]

Dual formulation of WRI - Lagrangian

$$\min_{\mathbf{m},\mathbf{u}} \frac{1}{2}\underbrace{||\mathbf{q} - A(\mathbf{m})\,\mathbf{u}||^2}_{\text{PDE misfit}} \quad \text{s.t.} \quad \underbrace{||\mathbf{d} - R\,\mathbf{u}|| \leq \varepsilon}_{\text{data constraint}}$$

Saddle-point problem:

$$\max_{\mathbf{y}} \min_{\mathbf{m},\mathbf{u}} \mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{y})$$

$$\mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{y}) = \frac{1}{2}||\mathbf{q} - A(\mathbf{m})\,\mathbf{u}||^2 + \langle \mathbf{y}, \mathbf{d} - R\,\mathbf{u}\rangle - \varepsilon||\mathbf{y}||$$

Dual formulation of WRI - Source focusing

$$\mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{y}) = -\frac{1}{2}||\mathbf{F}(\mathbf{m})^{*}\mathbf{y}||^{2}_{\Sigma^{-1}} + <\mathbf{y},\mathbf{d}-\mathbf{R}\mathbf{u}> -\epsilon||\mathbf{y}||$$

Where:

$$||\mathbf{p}||^{2}_{\Sigma^{-1}} = \sqrt{<\Sigma^{-1}\mathbf{p},\mathbf{p}>}$$ Weighted two norm

$$\Sigma^{-1/2} = \mathbf{diag}(\mathbf{w})$$

$$\mathbf{w} = \frac{\sqrt{(\mathbf{x}-\mathbf{x}_{s})^{2}+h^{2}}}{h}$$ Focusing at source position

Huang, G., Nammour, R., and Symes, W. W., 2017, Full-waveform inversion via source-receiver extension: Geophysics, **82**, R153–R171.

Huang, G., Nammour, R., and Symes, W. W., 2018, Volume source-based extended waveform inversion: Geophysics, **83**, R369–R387.

Dual formulation of WRI - Dual variable approximation

$$\mathcal{L}(\mathbf{m},\mathbf{u},\mathbf{y}) = -\frac{1}{2}||\mathbf{F}(\mathbf{m})^*\mathbf{y}||^2 + <\mathbf{y},\mathbf{r}(\mathbf{m})> -\epsilon||\mathbf{y}||$$

Project y out:

$$\tilde{y} = \tilde{\alpha}\mathbf{r}(\mathbf{m}),$$

$$\tilde{\alpha} = \begin{cases} \dfrac{\|\mathbf{r}(\mathbf{m})\|\left(\|\mathbf{r}(\mathbf{m})\| - \epsilon\right)}{\left\|F(\mathbf{m})^*\mathbf{r}(\mathbf{m})\right\|^2}, & \|\mathbf{r}(\mathbf{m})\| \geq \epsilon, \\ 0, & \text{otherwise.} \end{cases}$$

Leads to:

$$\nabla_{\mathbf{m}}\tilde{\mathcal{L}}_{\mathrm{WRI}^*} = \nabla_{\mathbf{m}}\mathcal{L}_{\mathrm{WRI}^*}(\mathbf{m},\tilde{\mathbf{y}}) - \tilde{\alpha}J[\mathbf{m},\mathbf{q}]^*\nabla_{\mathbf{y}}\mathcal{L}_{\mathrm{WRI}^*}(\mathbf{m},\tilde{\mathbf{y}})$$

Traditional WRI

$$A(\mathbf{m}) \Leftrightarrow \quad \frac{1}{c^2}\frac{d^2p(x,t)}{dt^2} - \Delta p(x,t) = 0$$

Easy to solve with Helmholtz

Scalar equation

Single parameter

Self-adjoint

Cheap $\mathcal{O}(10)$ FLOPS/gridpoint

Real world physics

$$\frac{\partial \mathbf{v}}{\partial t} = \nabla . \tau$$

Vectorial, 3 components

$$\frac{\partial \tau}{\partial t} = \mathcal{C} :: e \quad \text{where: } e = \frac{1}{2}(\nabla \mathbf{v} + (\nabla \mathbf{v})^T)$$

Symmetric tensorial, 9 components
21 independent parameters in C

Not feasible computationally => physical approximation:

- Acoustic isotropic , too simple
- Elastic isotropic , 3 parameters, still vectorial and elastic (elastic = 16X acoustic)
- Pseudo-acoustic anisotropic (TTI). Non-physical but feasible and accurate enough

Transverse tilted anisotropic wave equation (TTI)

$$A(\mathbf{m}) \Leftrightarrow \begin{aligned} &m(x)\frac{d^2p(x,t)}{dt^2} - (1+2\epsilon(x))H_{\bar{x}\bar{y}}p(x,t) - \sqrt{1+2\delta(x)}\; H_{\bar{z}}r(x,t) = 0, \\ &m(x)\frac{d^2r(x,t)}{dt^2} - \sqrt{1+2\delta(x)}\; H_{\bar{x}\bar{y}}p(x,t) - H_{\bar{z}}r(x,t) = 0 \end{aligned}$$

Non trivial in frequency domain

Expensive $\mathcal{O}(1000)$ FLOP/gridpoint (rotated anisotropic laplacian)

Non self-adjoint from physical approximation

$$A^\top(\mathbf{m}) \Leftrightarrow \begin{aligned} &m(x)\frac{d^2p_a(x,t)}{dt^2} - H_{\bar{x}\bar{y}}(1+2\epsilon(x))p_a(x,t) - H_{\bar{x}\bar{y}}\sqrt{1+2\delta(x)}r_a(x,t) = 0, \\ &m(x)\frac{d^2r_a(x,t)}{dt^2} - H_{\bar{z}}\sqrt{1+2\delta(x)}p_a(x,t) - H_{\bar{z}}r_a(x,t) = 0 \end{aligned}$$

Dual formulation of WRI - Computational notes

Algorithm:

1. Forward wavefield $\mathbf{u} = A(\mathbf{m})^{-1}\mathbf{q}$ and residual $\mathbf{r} = \mathbf{d} - R\mathbf{u}$
2. Backward wavefield $\tilde{\mathbf{q}} = A(\mathbf{m})^{-*}R^*\mathbf{r}(\mathbf{m})$
3. Compute $\alpha = \alpha(\|\mathbf{r}\|, \|\tilde{\mathbf{q}}\|, \epsilon)$, equation (19)
4. Objective value $L = -\alpha^2\|\tilde{\mathbf{q}}\|^2/2 + \alpha\|\mathbf{r}\|^2 - |\alpha|\epsilon\|\mathbf{r}\|$, equation (20)
5. Augmented wavefield $\tilde{\mathbf{u}} = A(\mathbf{m})^{-1}(\mathbf{q} + \alpha\tilde{\mathbf{q}})$ and residual $\tilde{\mathbf{r}} = \mathbf{d} - R\tilde{\mathbf{u}}$
6. Gradient $\mathbf{g}_{\mathbf{m},1}(\mathbf{x}) = -\alpha\sum_t \partial_{tt}\tilde{\mathbf{u}}(\mathbf{x},t)\tilde{\mathbf{q}}(\mathbf{x},t)$, equation (15)
7. Gradient $\mathbf{g}_{\mathbf{y}} = \tilde{\mathbf{r}} - \text{sign}(\alpha)\,\epsilon\,\mathbf{r}/\|\mathbf{r}\|$, equation (16)
8. Backpropagated gradient $\mathbf{v} = A(\mathbf{m})^{-*}R^*\mathbf{g}_{\mathbf{y}}$
9. Gradient correction $\mathbf{g}_{\mathbf{m},2} = -\alpha\sum_t \partial_{tt}\mathbf{u}(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)$
10. Corrected gradient $\mathbf{g}_{\mathbf{m}} = \mathbf{g}_{\mathbf{m},1} + \mathbf{g}_{\mathbf{m},2}$, equation (21)

Output: L,$\mathbf{g}_{\mathbf{m}}$

PDE solve count

WRI	FWI
1	1
1	0
1	0
1, 0	1
4, 3	2

2X FWI , 1.5X FWI without optimal alpha

Implementation

[1] P. A. Witte, M. Louboutin, N. Kukreja, F. Luporini, M. Lange, G. G. Gorman and Felix J. Herrmann, 2019, *A large-scale framework for symbolic implementations of seismic inversion algorithms in Julia*, Geophysics, 84, 3.
[2] M. Louboutin, M. Lange, F. Luporini, N. Kukreja, P. A. Witte, F. J. Herrmann, P. Velesko and G. J. Gorman, 2019, Devito 3.1.0: an embedded domain-specific language for finite differences and geophysical exploration: Geoscientific model development, 12, 3.
[3] F. Luporini, M. Lange, M. Louboutin, N. Kukreja, J. Hückelheim, , C. Yount, P. A. Witte, P. H. J. Kelly, F. J. Herrmann and G. J. Gorman, 2019, Architecture and performance of Devito, a system for automated stencil computation, arXiv preprints.

The Julia Devito Inversion framework (JUDI)

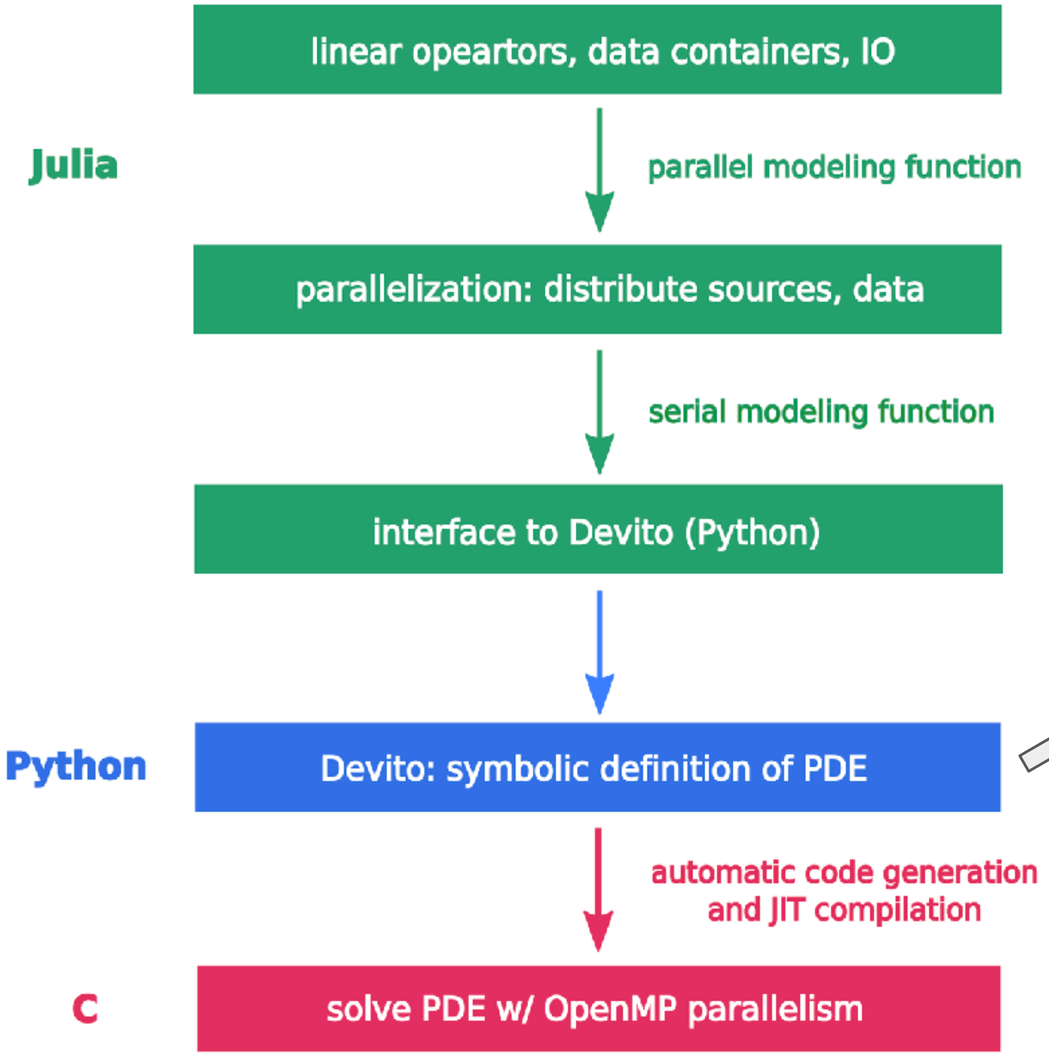
JUDI.jl:[1]

- matrix-free linear operators and abstract data containers
- parallel I/O based on look-up tables, parallelism (OMP + soon MPI)
- interface to Julia's machine learning package Flux
- *https://github.com/slimgroup/JUDI.jl/* (MIT license)

Devito:[2][3]

- a domain-specific language for finite-differences in Python
- code generation with automated performance optimizations
- model parallelism (MPI, multithreading)
- *https://github.com/devitocodes/devito* (MIT license)

The Julia Devito Inversion framework (JUDI)




```
g = J'*(Pr*A_inv*Ps'*q - d_obs)
```

```
# Symbolic wavefield
u = TimeFunction(name='u', space_order=16)

# Acoustic wave equation
pde = model.m * u.dt2 - u.laplace
```

```
float (*rec_coords)[2] = (float (*)[2]) rec_coords_vec;
{
  struct timeval start_kernel, endkernel;
    gettimeofday(&start_kernel, NULL);
    int t0;
    int t1;
    int t2;
    ;
    for (int i3 = 0; i3 < 3; i3+=1)
    {
      flops->kernel += 2.000000;
      {
        {
          t0 = (i3)%(3);
```

Devito

15 Lines of code

Based on generic propagators

Discretization and physics flexibility

```python
def wri_func(model, src_coords, wavelet, rec_coords, recin, yin, space_order=8,
             isic=False, ws=None, t_sub=1, grad="m", grad_corr=False,
             alpha_op=True, w_fun=None, eps=0):
    """
    Time domain wavefield reconstruction inversion wrapper
    """
    # F(m0) * q if y is not an input and compute y = r(m0)
    y, u0, _ = forward(model, src_coords, rec_coords, wavelet, save=grad_corr,
                       space_order=space_order, ws=ws)
    y = recin[:] - y.data[:]
    # Compute wavefield vy = adjoint(F(m0))*y and norm on the fly
    srca, v, norm_v, _ = adjoint(model, y, src_coords, rec_coords,
                                 norm_v=True, w_fun=w_fun,
                                 save=grad is not None)
    # <PTy, d-F(m)*f> = <PTy, d> - <adjoint(F(m))*PTy, f>
    PTy_dot_r = model.critical_dt * (np.dot(y(-1), recin(-1)) -
                                     np.dot(srca(-1), wavelet(-1)))
    norm_y = np.sqrt(model.critical_dt) * np.linalg.norm(y)
    # alpha
    alpha = compute_optalpha(norm_y, norm_v, eps, comp_alpha=alpha_op)
    # Lagrangian evaluation
    phi = -.5 * alpha**2 * norm_v + alpha * PTy_dot_r -
         eps * np.abs(alpha) * norm_y
    # Focusing function
    w = weight_fun(w_fun, model, src_coords)
    w = c1*alpha/w**2 if w is not None else c1*alpha
    Q = wf_as_src(v, w=w)
    rcv, gradm, _ = forward_grad(model, src_coords, rec_coords, wavelet, v, q=Q)
    # Compute gradient wrt y
    grady = recin - rcv.data[:] - np.abs(eps) * ydat / norm_y
    # Correcting for reduced gradient
    gradm_corr, _ = gradient(model, grady, rec_coords, u0)
    gradm = gradm.data + gradm_corr.data
    return phi, alpha * gradm, grady
```

Generic propagators

Solution wavefield (any space order)

Time-space source (any space order)

PDE (flexible physics)

Source/receivers

Code generation, JIT

```python
# Forward propagation
def forward(model, src_coords, rcv_coords, wavelet, space_order=8
            q=None, return_op=False, freq_list=None, dft_sub=None
            ws=None, t_sub=1, **kwargs):
    """
    Low level propagator, to be used through `interface.py`
    Compute forward wavefield u = A(m)^{-1}*f and
    related quantities (u(xrcv))
    """
    # Number of time steps
    nt = wavelet.shape[0]
    # Setting forward wavefield
    u = wavefield(model, space_order, save=save, nt=nt, t_sub=t_s
   # Add extended source
    q = extented_src(model, ws, wavelet, q=q or wf_as_src(u, w=0)
    # Set up PDE expression and rearrange
    pde, tmp = wave_kernel(model, u, q=q)
    # Setup source and receiver
    geom_expr, _, rcv = src_rec(model, u, src_coords=src_coords,
                                rec_coords=rcv_coords, wavelet=wa
    # Create operator and run
    subs = model.spacing_map
    op = Operator(tmp + pde + geom_expr + dft + eq_save,
                  subs=subs, name="forward"+name(model),
                  opt=opt_op(model))
    summary = op()
    return rcv, (u_save if t_sub > 1 else u), summary
```

Modular framework

Workload based (PDE, geometry, model, interface, …)

Easy extensions and generalisations

Usable as standalone python as well

JUDI interface

```
optwri = TWRIOptions(;grad_corr=false, comp_alpha=false,
                      weight_fun=weight_fun_pars, eps=ε,params=:m)

fun, g = twri_objective(model0, fsrc, dat, nothing;
                        optionswri=optwri)
```

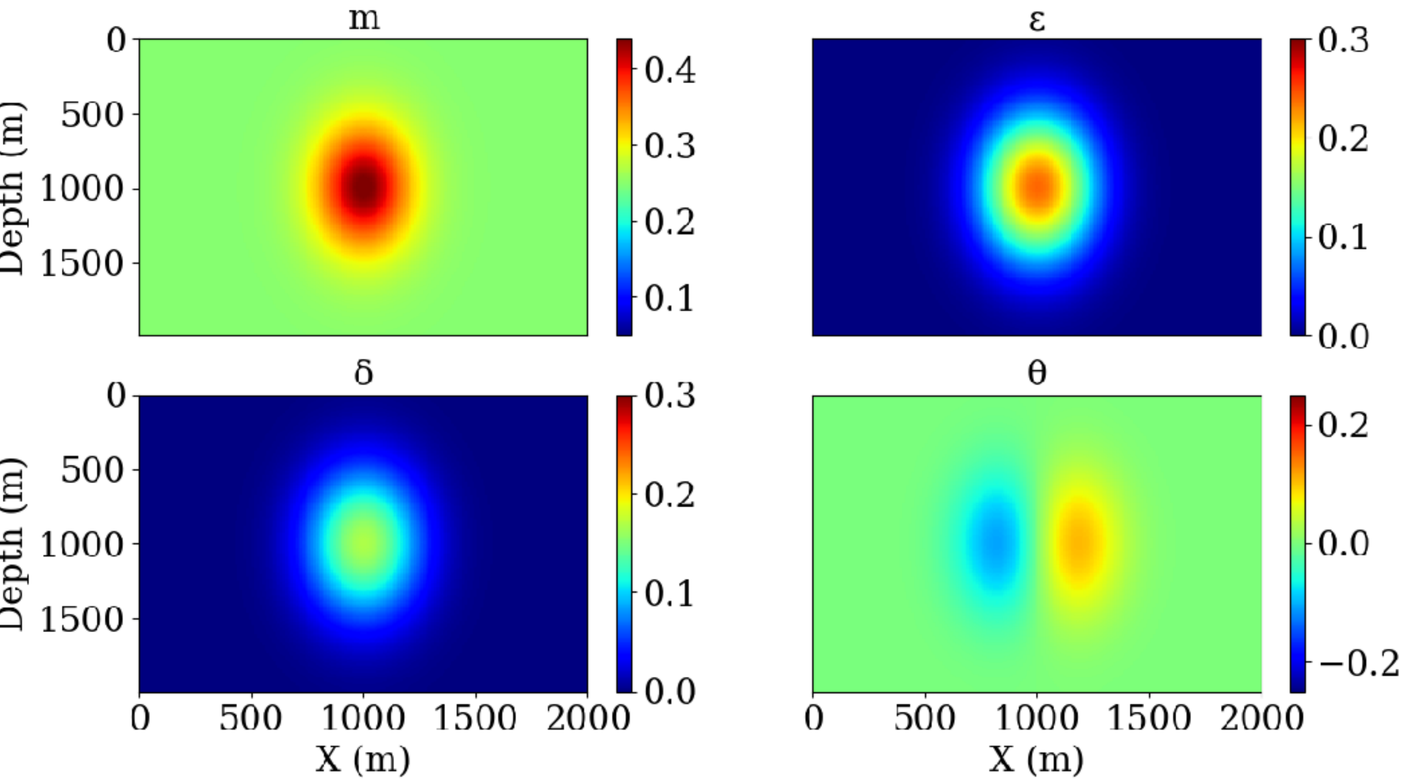
Builds on existing JUDI advantages:

- Julia task parallelisation
- SEGY IO (seismic data format)
- High performance Devito kernels (just-in-time code generation)
- Multiple physics (Isotropic acoustic, TTI)
- Open Source
- Continuous Integration

Numerical Experiments

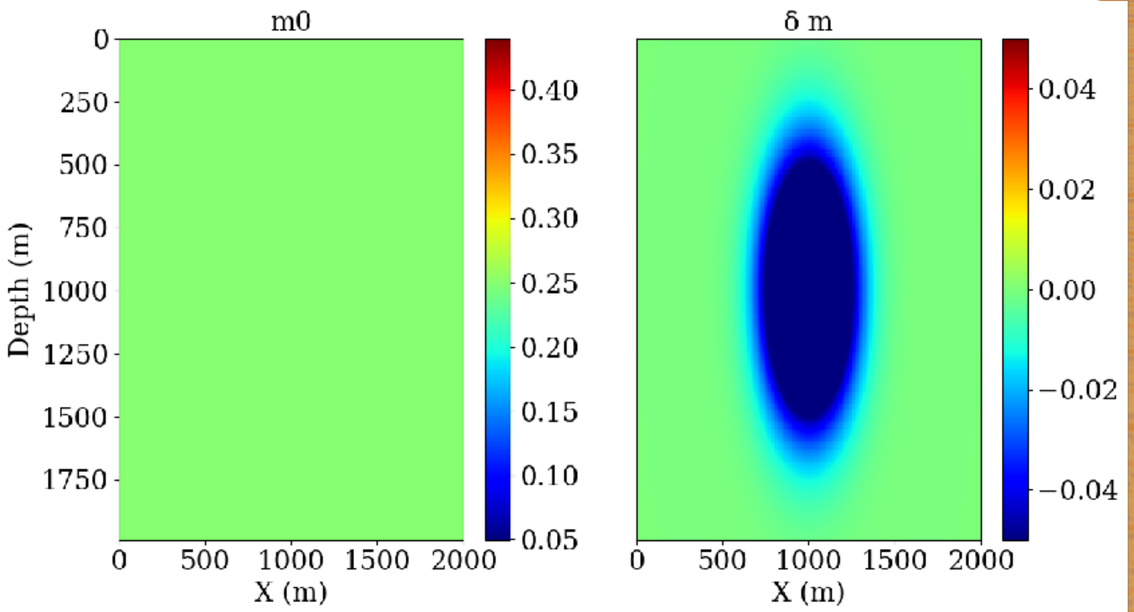
Gaussian lens

TTI lens model

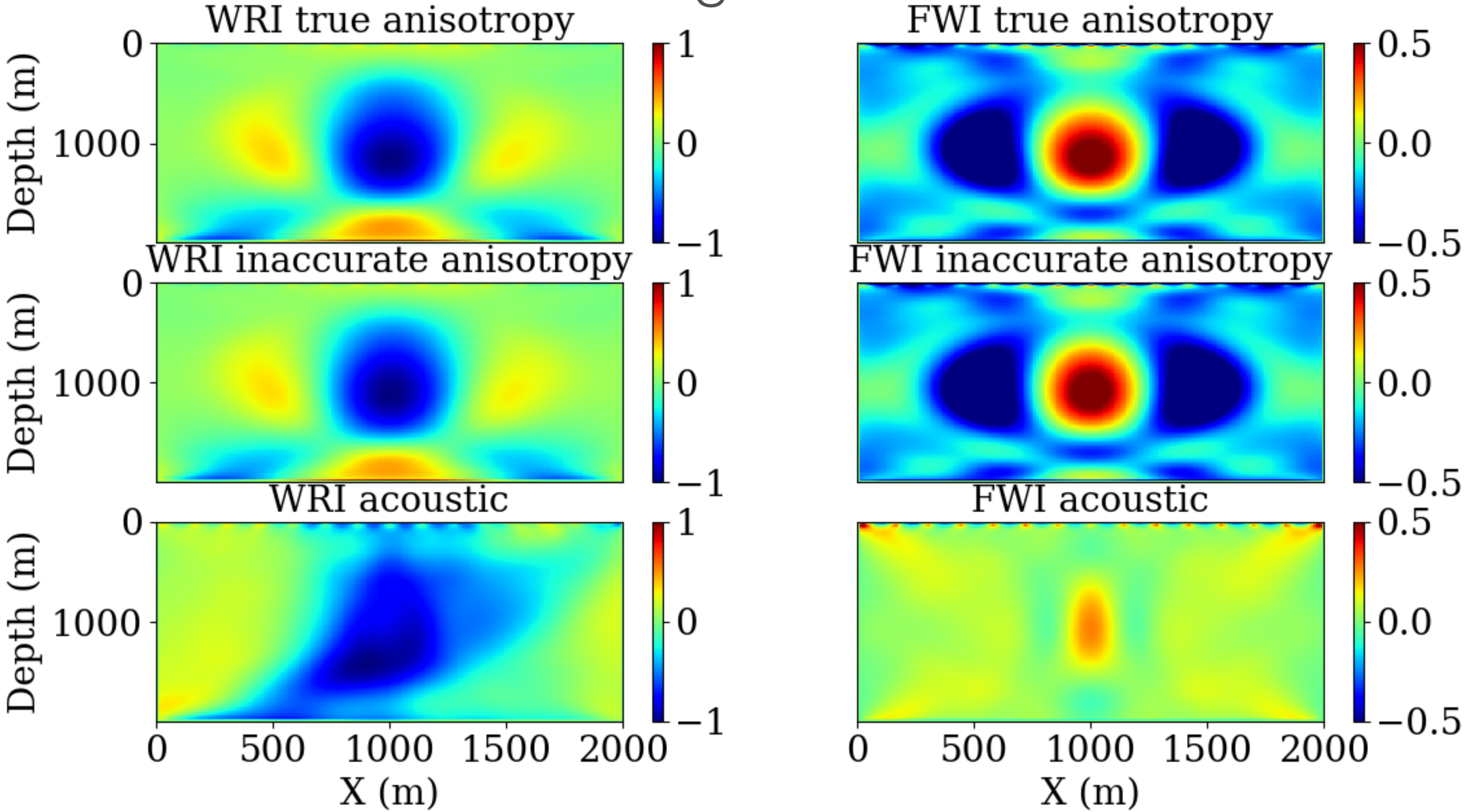


True model

Initial model



first gradient



Numerical Experiments

BG Compass

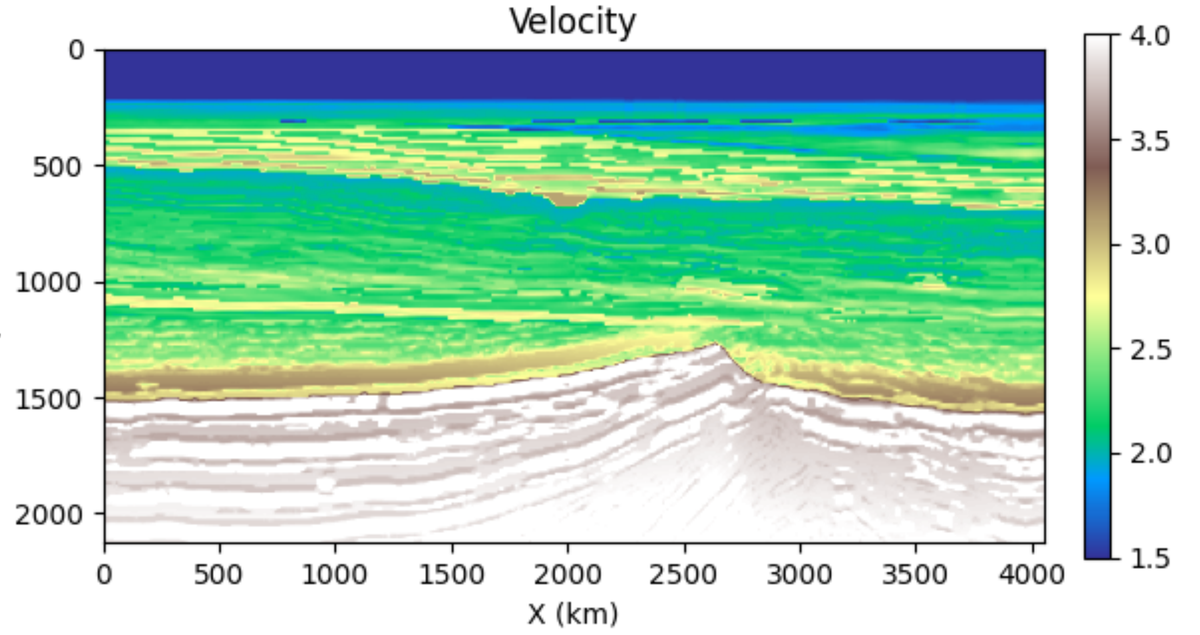
Schmidt, M., E. van den Berg, M. Friedlander, and K. Murphy, 2009, Optimizing costly functions with simple constraints: A limited-memory projected Quasi-Newton algorithm: Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, 456–463.

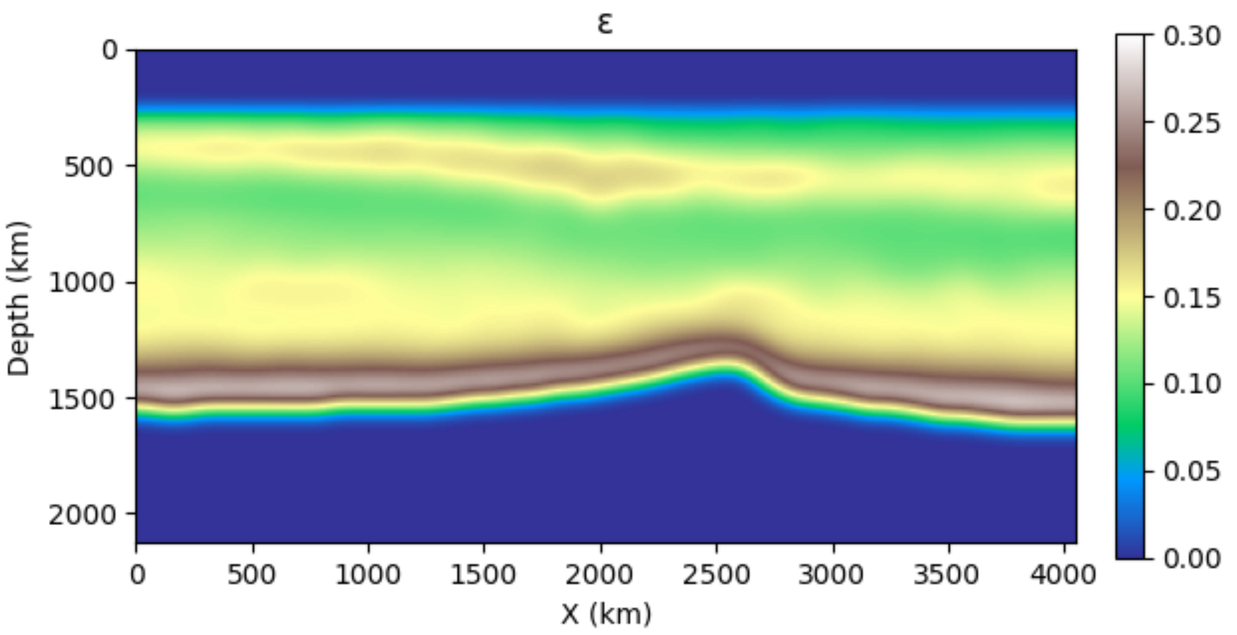
Algorithm

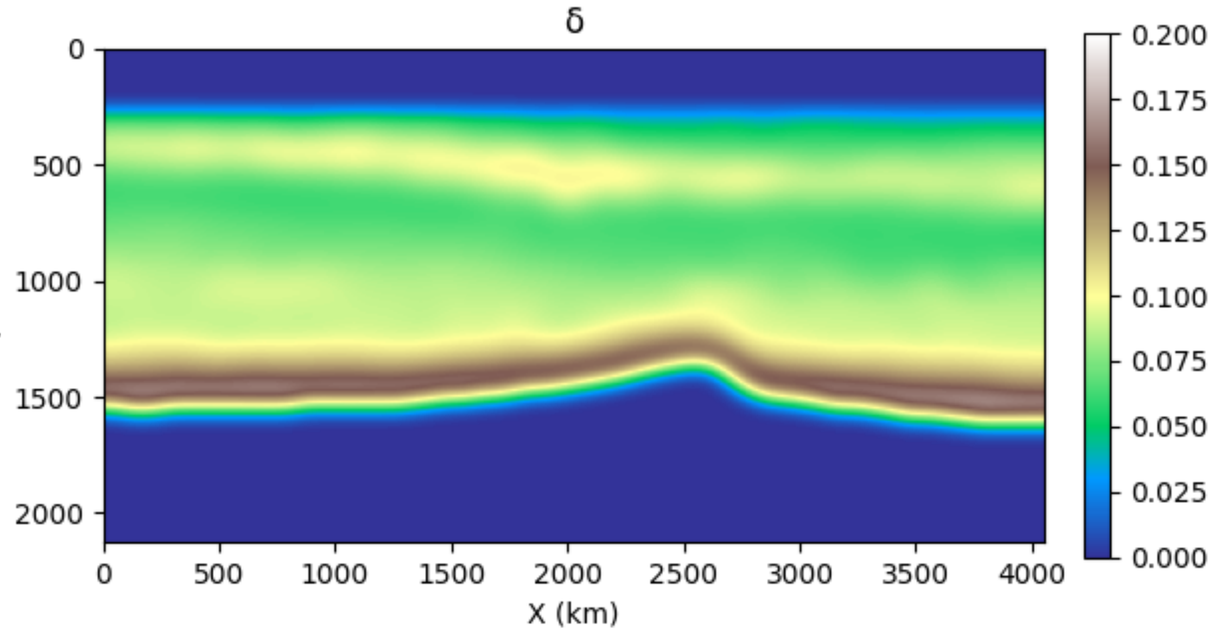
Fixed bandwidth centred at 10Hz

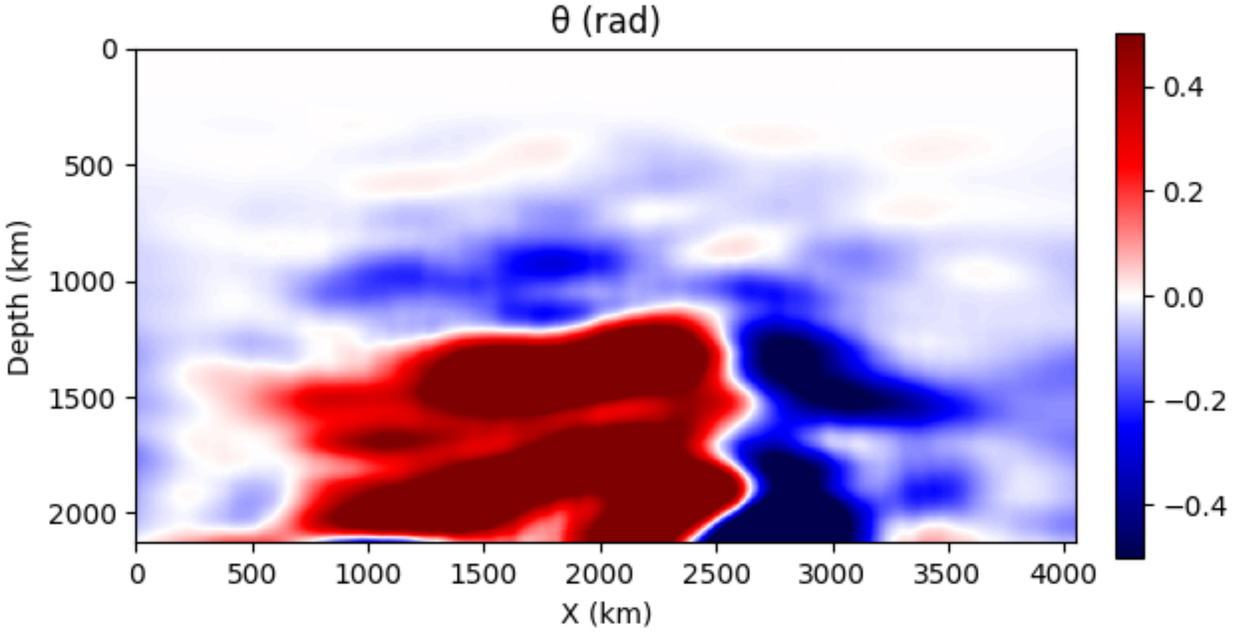
20 Iterations of SPG

Box constraints


Velocity
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)
4.0
3.5
3.0
2.5
2.0
1.5

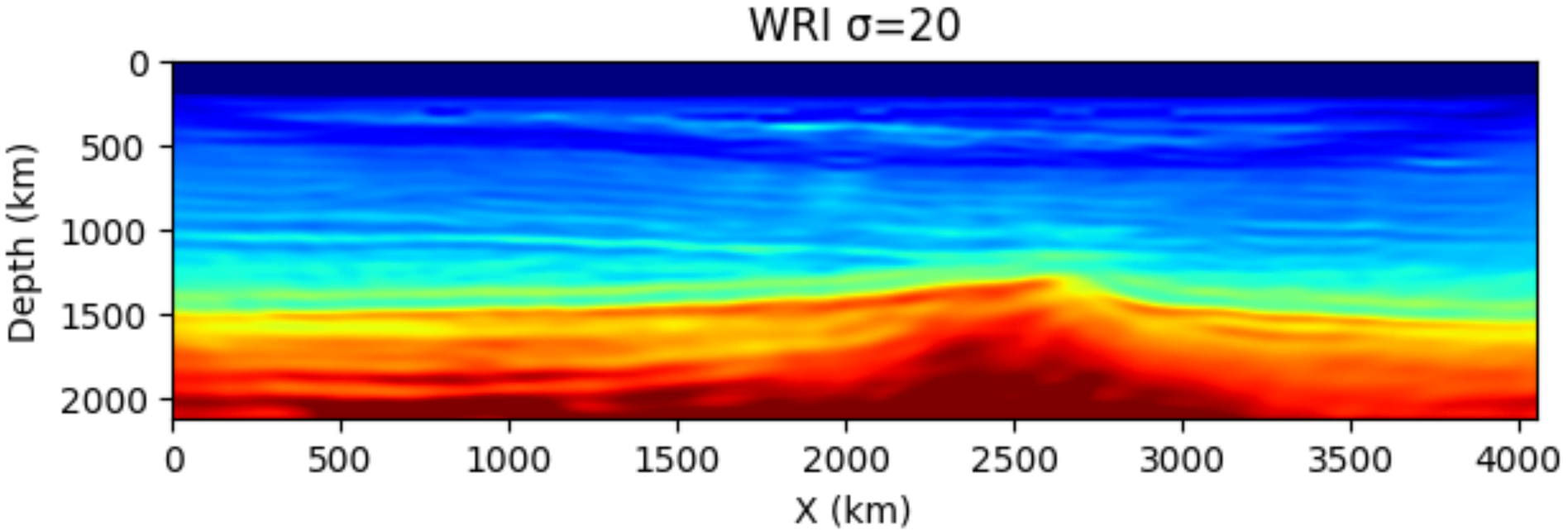

ε
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)
0.30
0.25
0.20
0.15
0.10
0.05
0.00

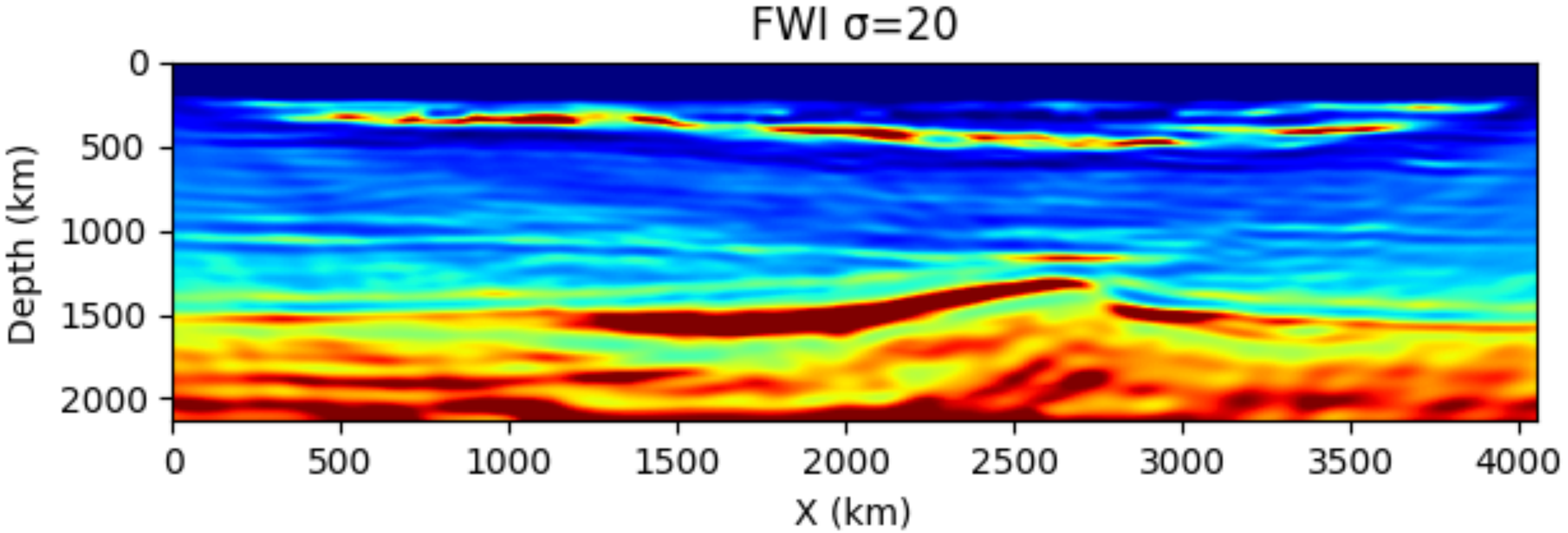

δ
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000

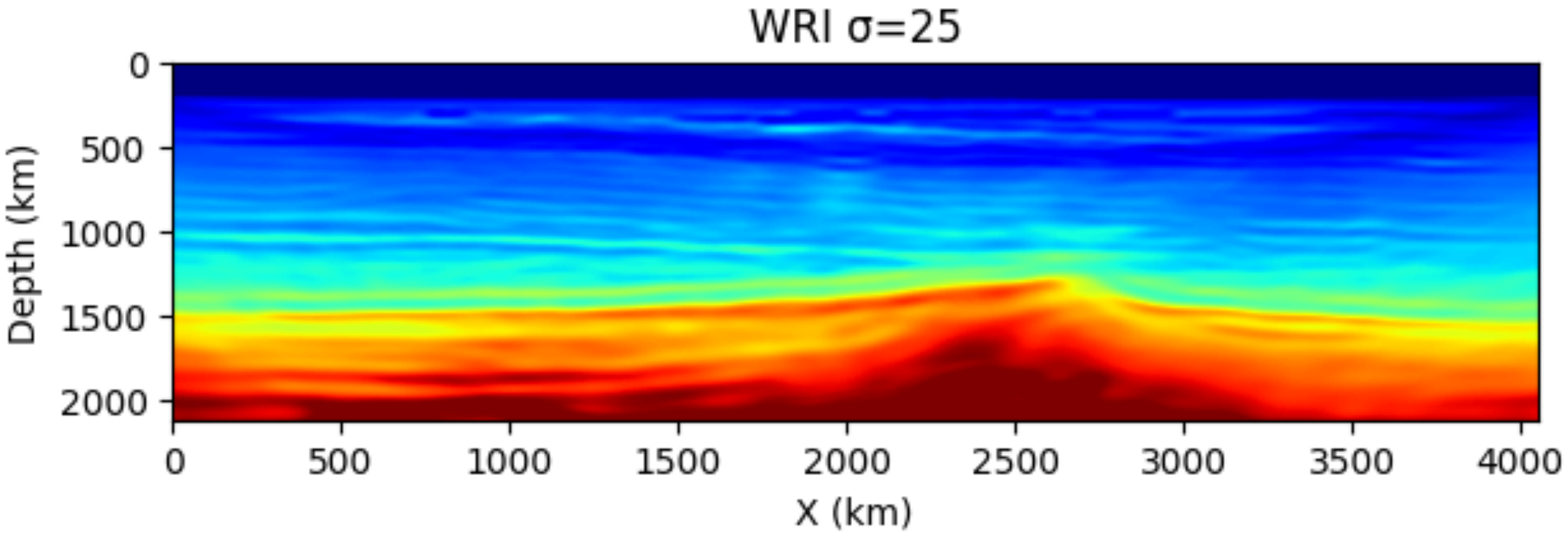

θ (rad)
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)
0.4
0.2
0.0
−0.2
−0.4

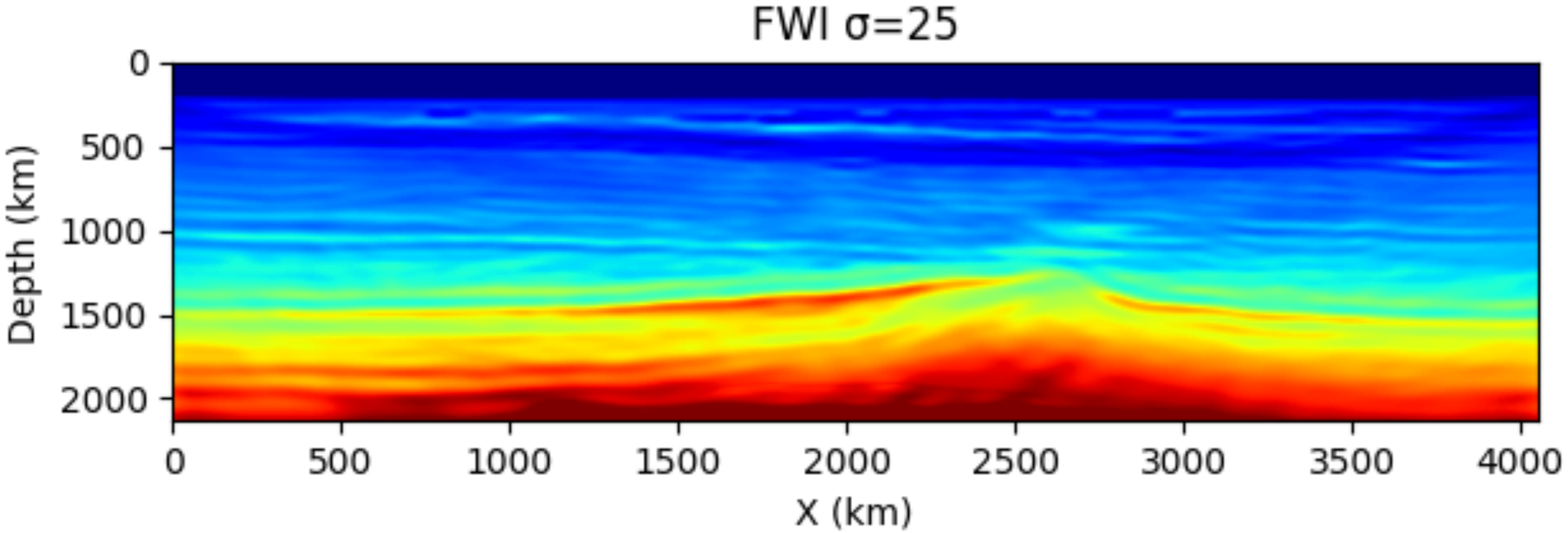
Inversion with true anisotropy

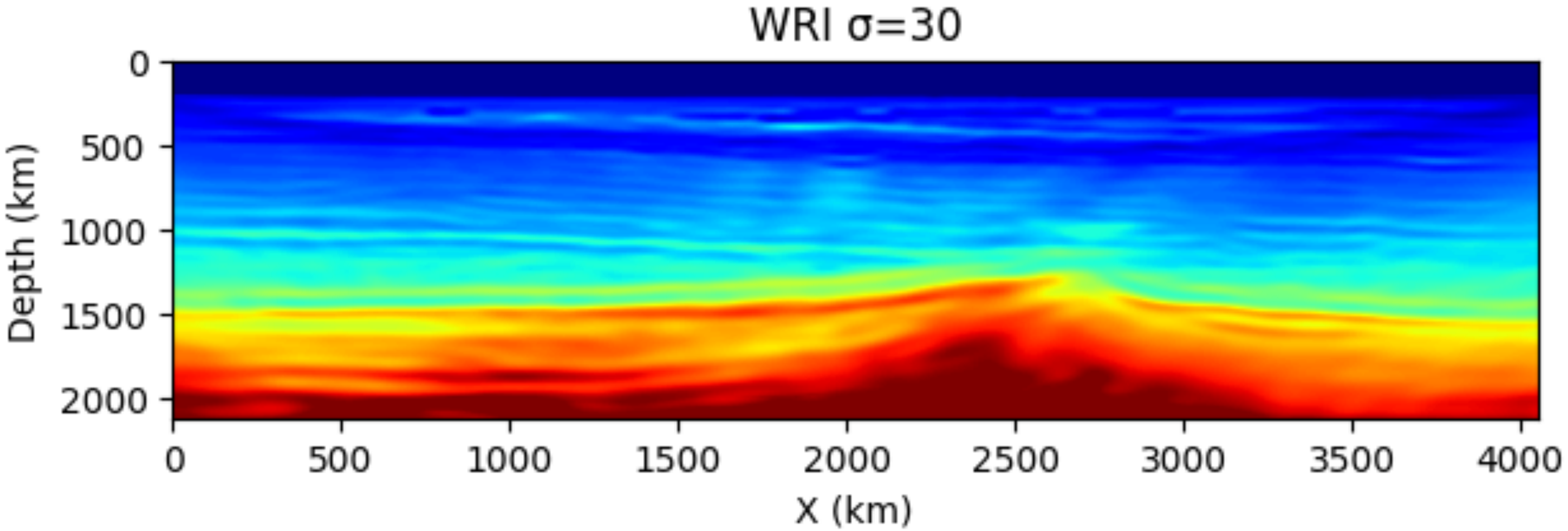
sigma: Gaussian kernel width for initial model

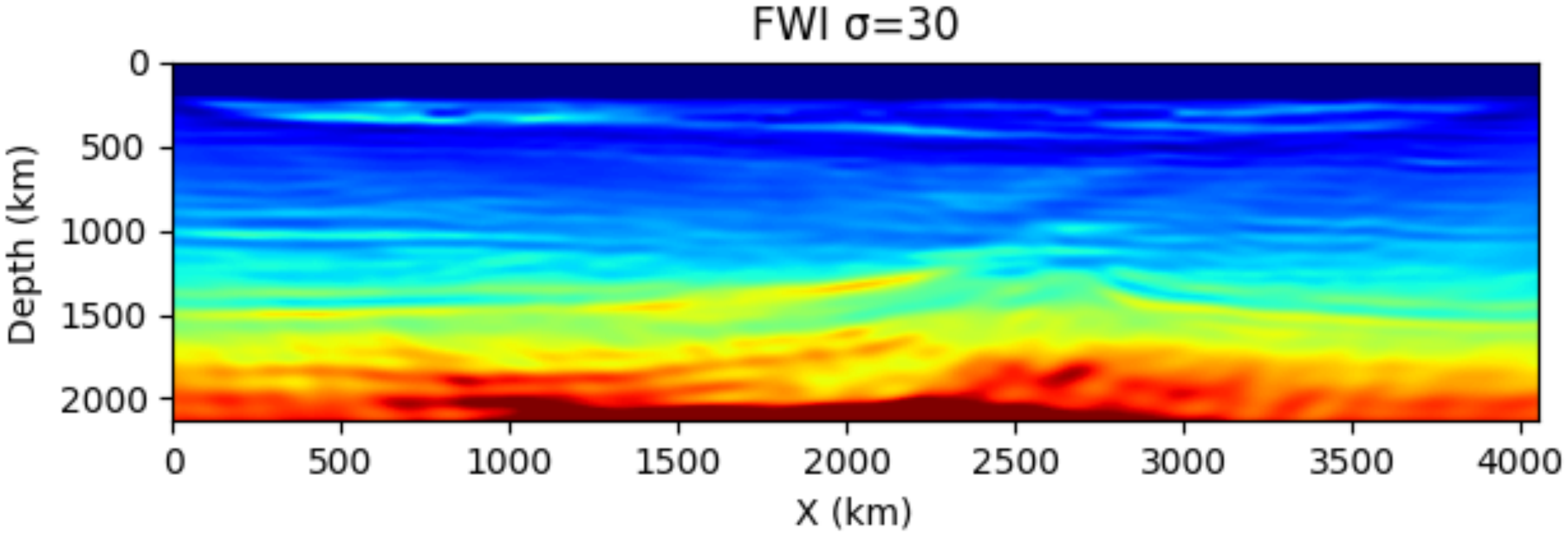

WRI σ=20
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

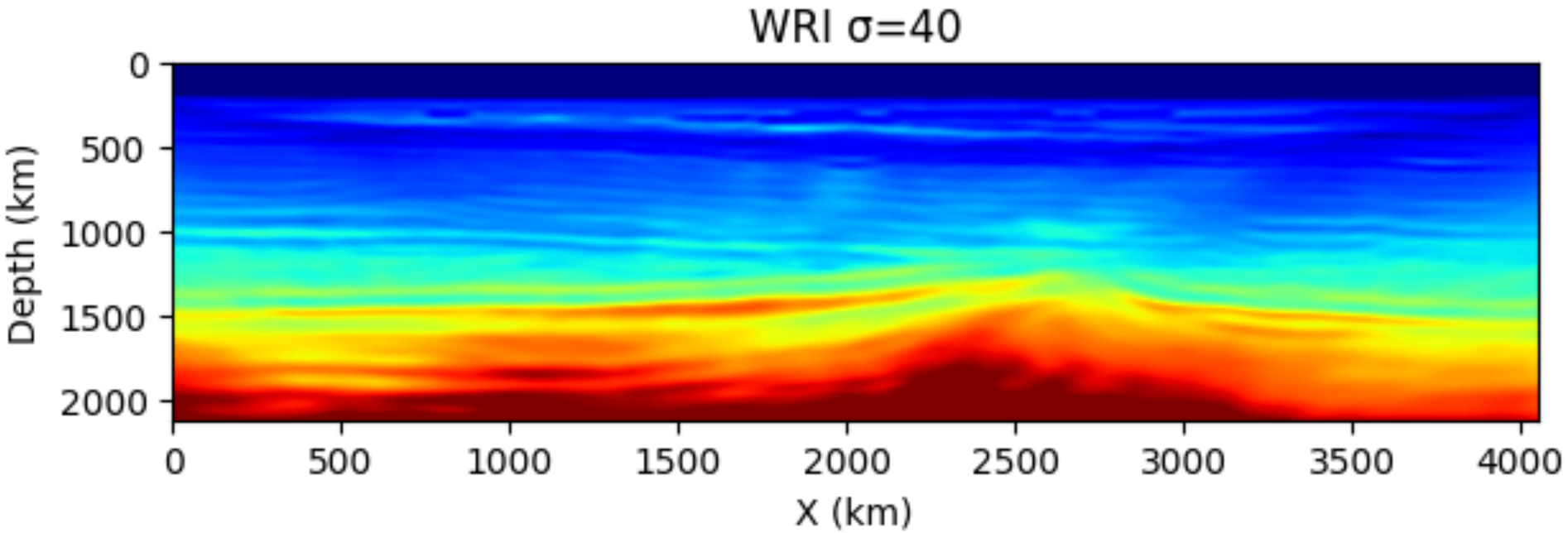

FWI σ=20
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

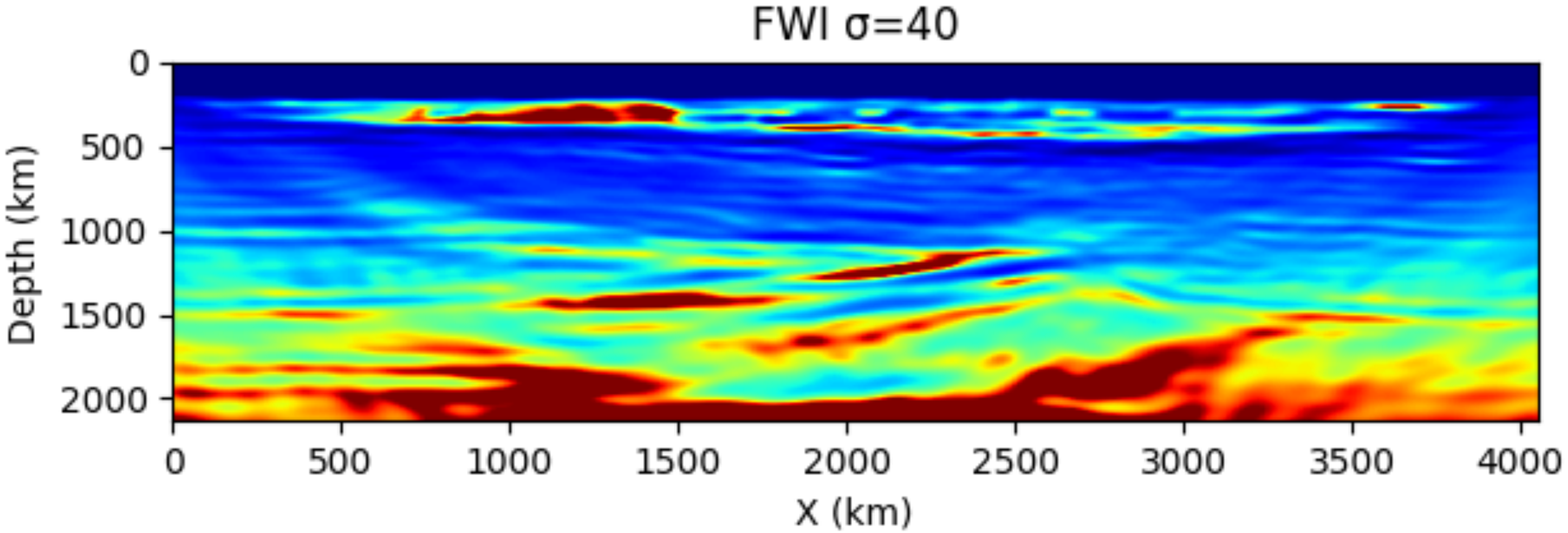

WRI σ=25
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=25
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


WRI σ=30
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=30
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

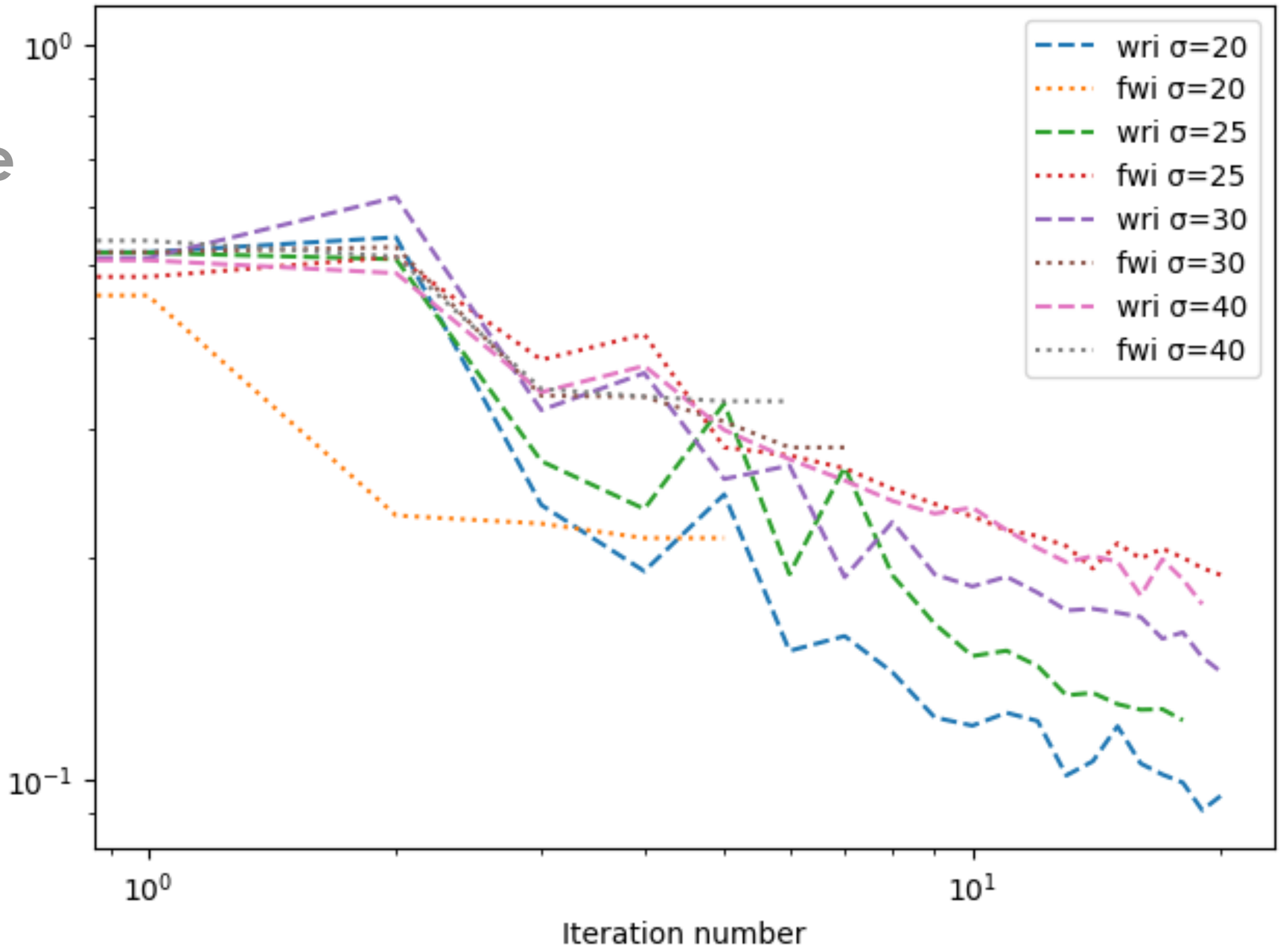

WRI σ=40
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=40
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

Convergence

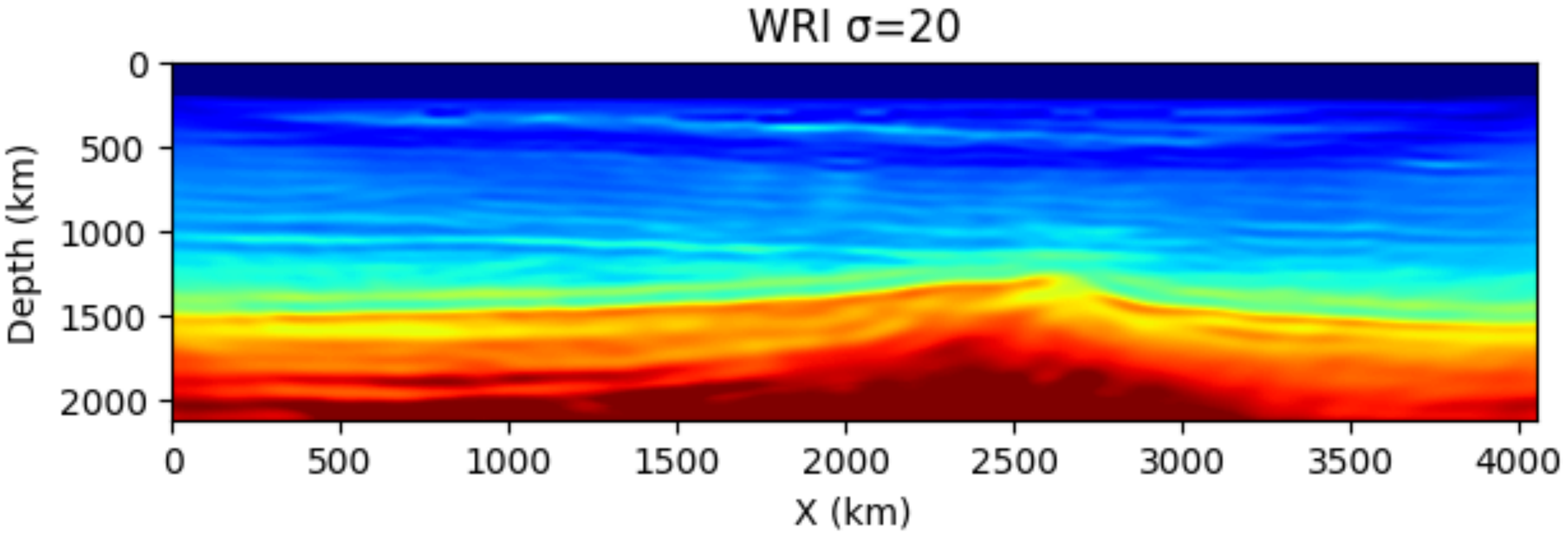
Lower misfit

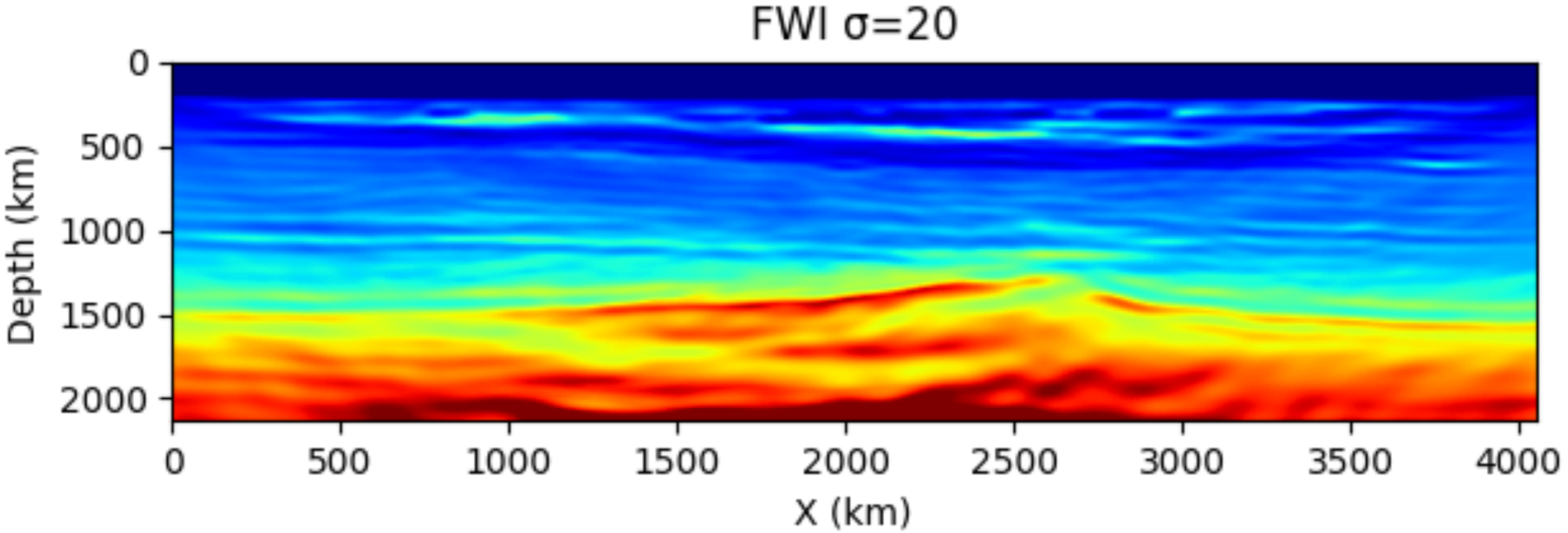
No plateau

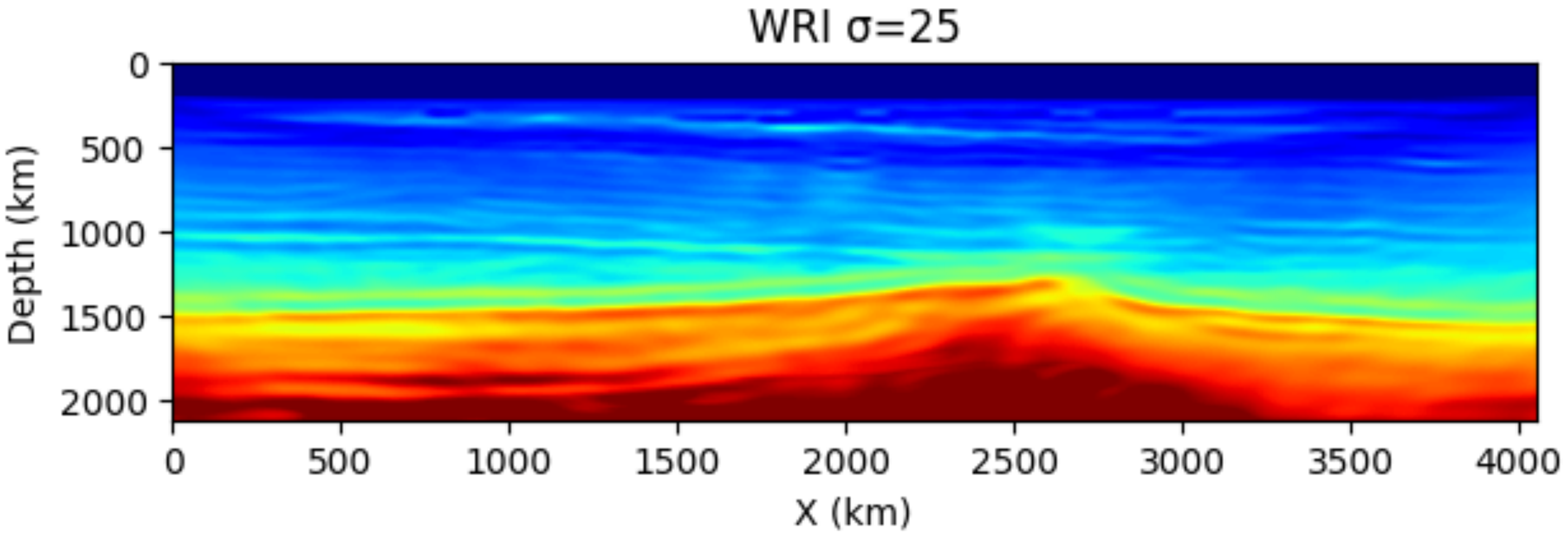


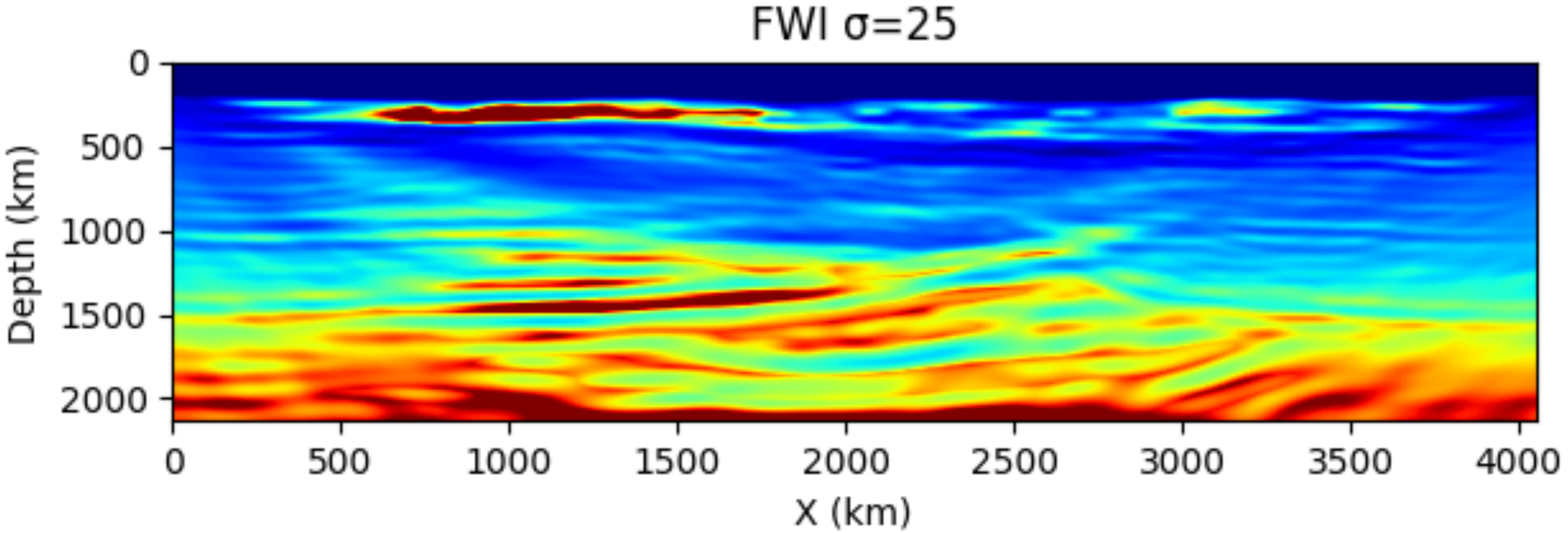
Inversion with inaccurate anisotropy

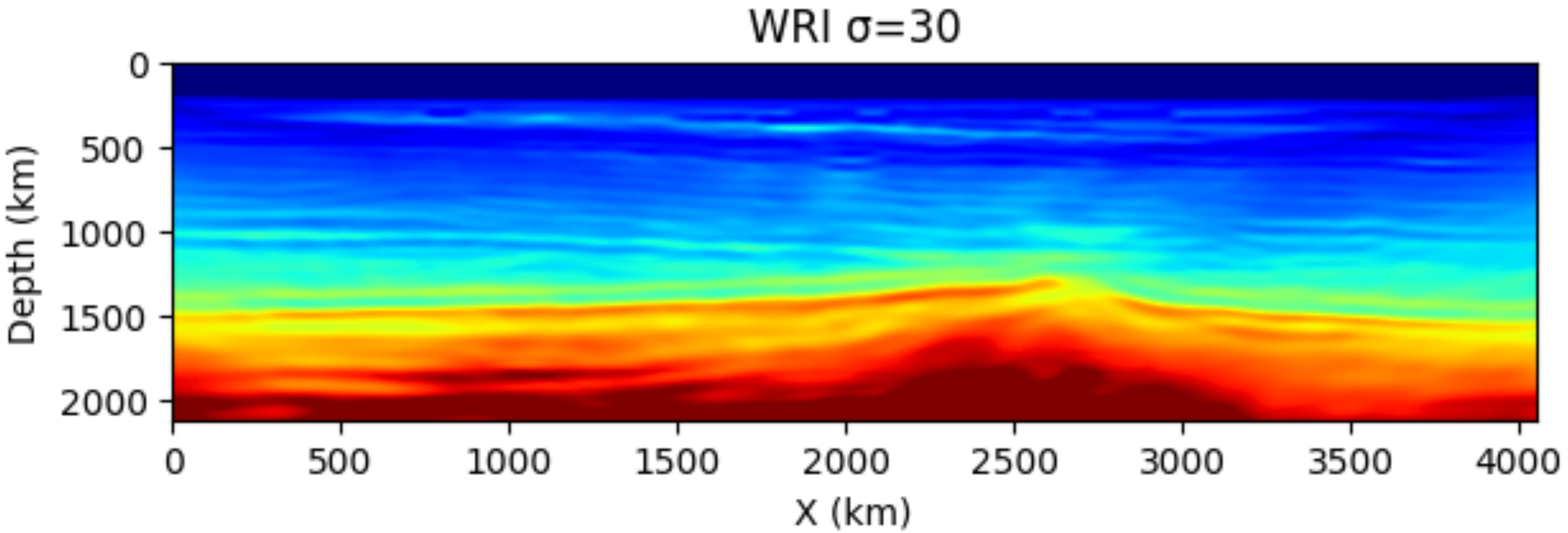
Anisotropy parameters smoothed with gaussian kernel.

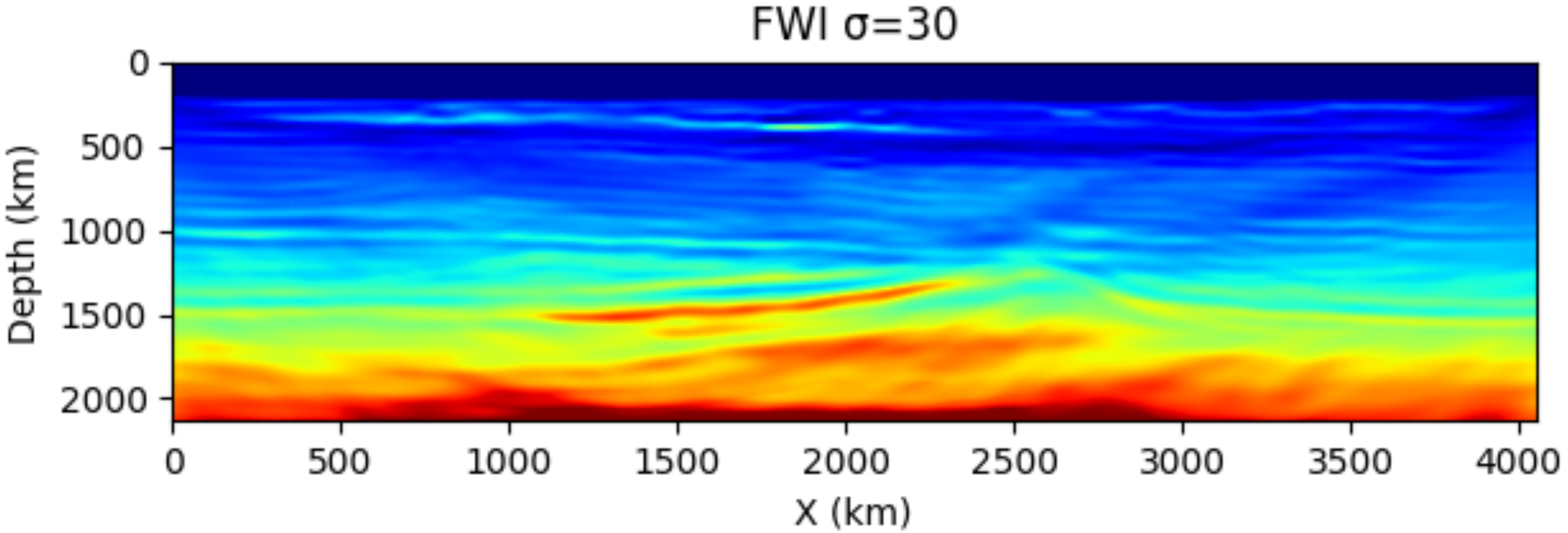

WRI σ=20
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

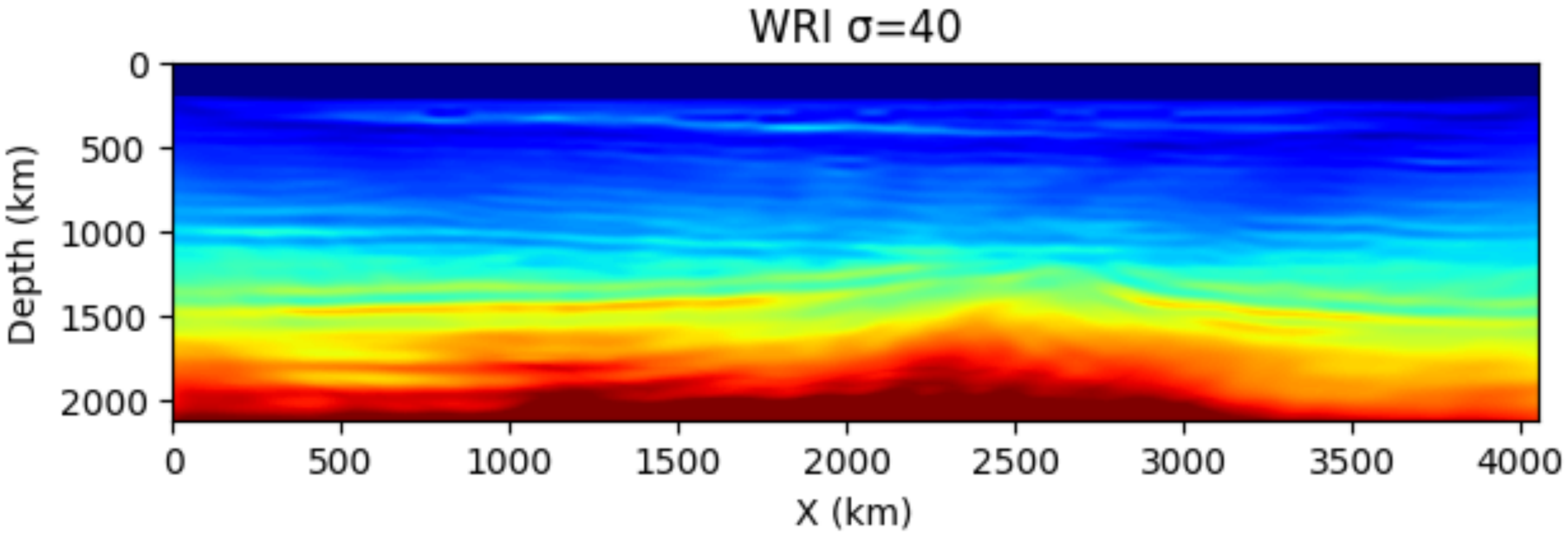

FWI σ=20
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

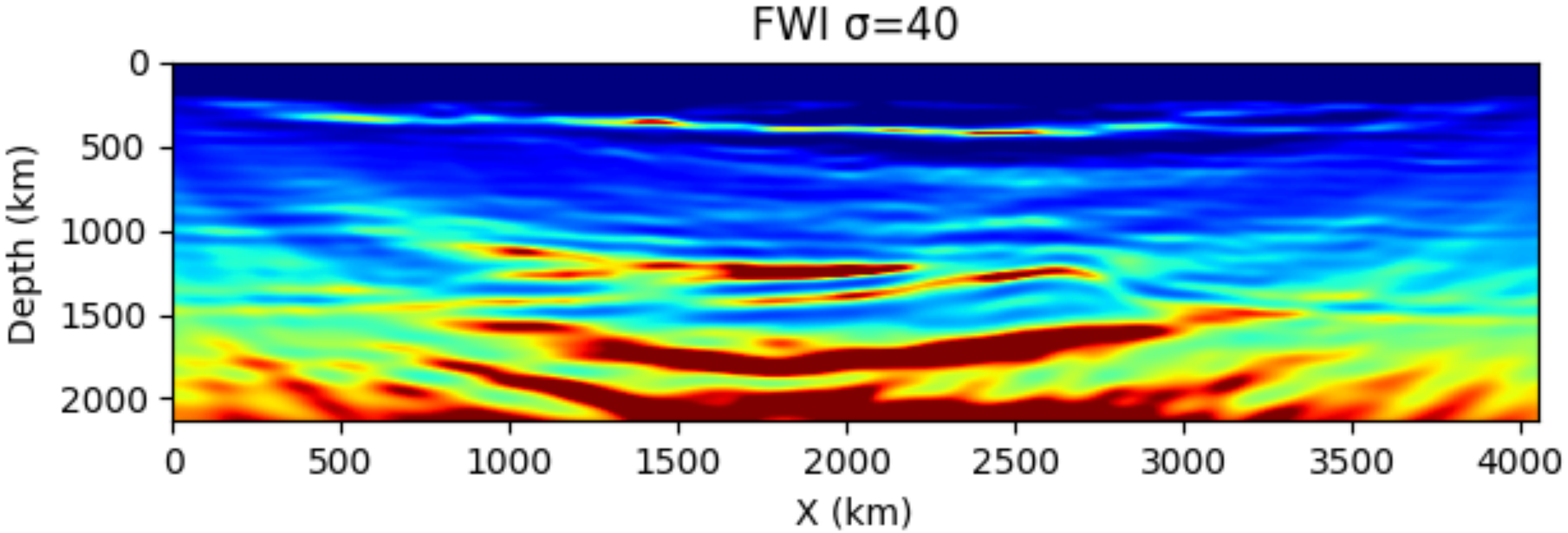

WRI σ=25
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=25
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


WRI σ=30
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=30
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

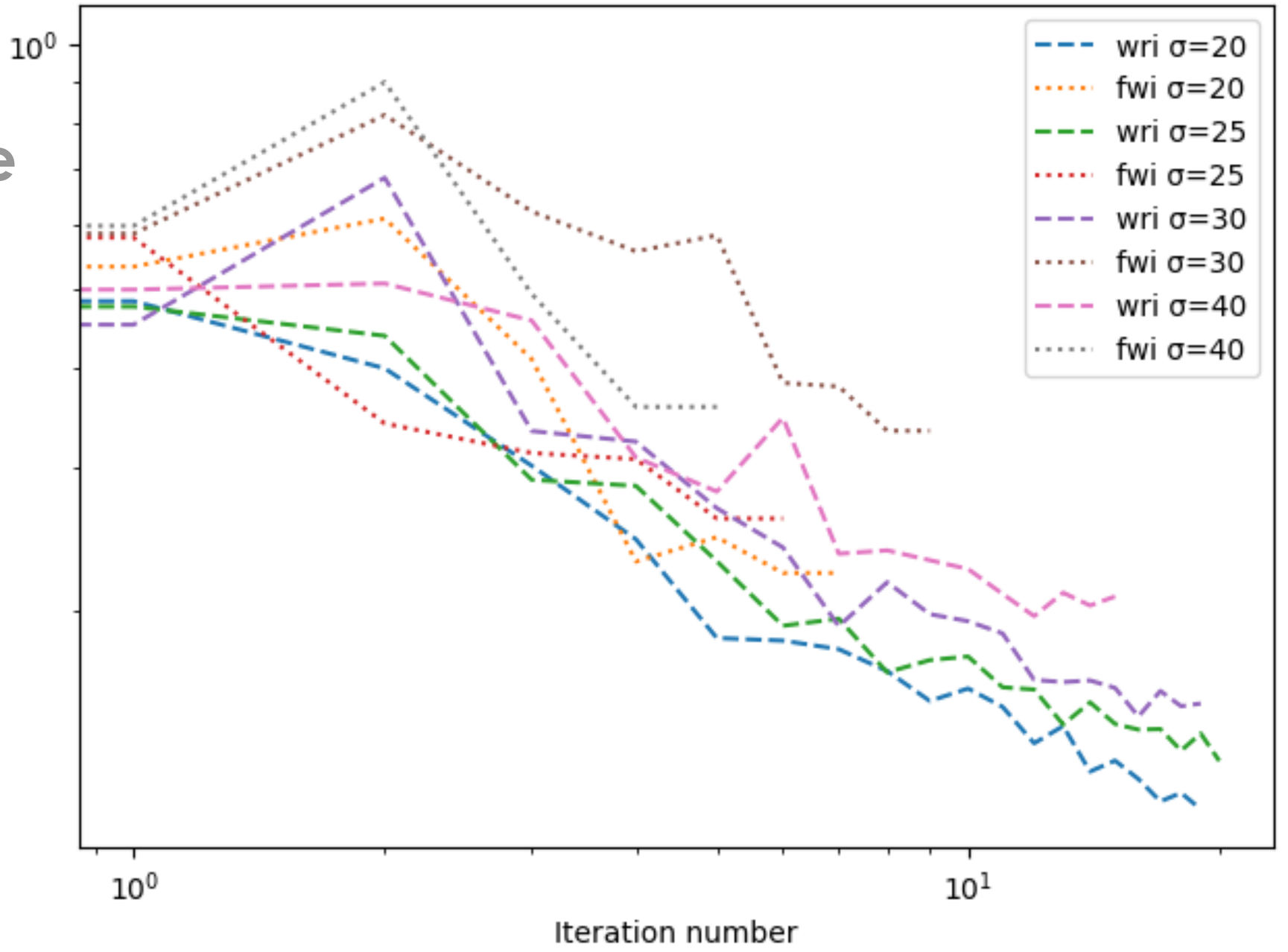

WRI σ=40
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)


FWI σ=40
Depth (km)
0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000
X (km)

Convergence

Lower misfit

No linesearch fail



Conclusions

True time-domain formulation

Inherit robustness of original formulation

Extends easily to realistic physical representations (TTI)

Computationally efficient through JUDI and Devito

Future work

3D

Elastic wave equation

Improve computational/memory cost

Preconditioning

3D first steps

Tim-domain enables 3D

3D TTI gradient

10kmx7.5kmx3km

10 sources