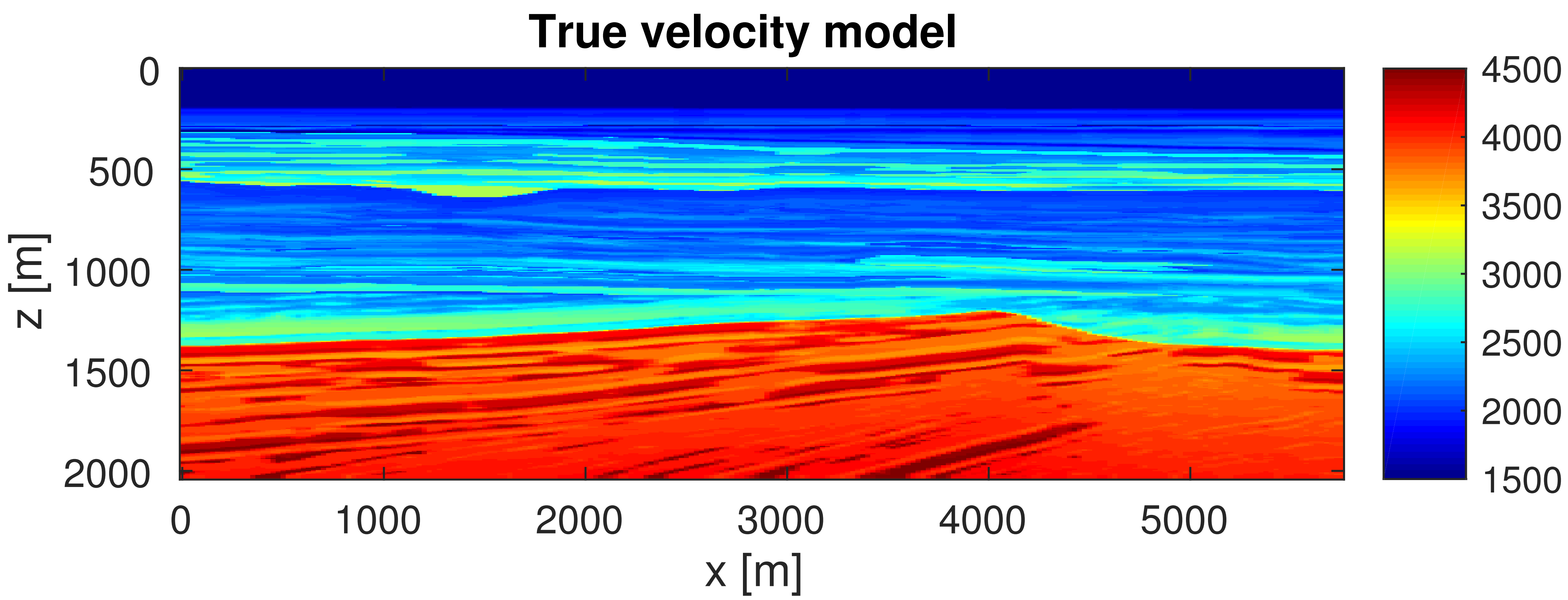
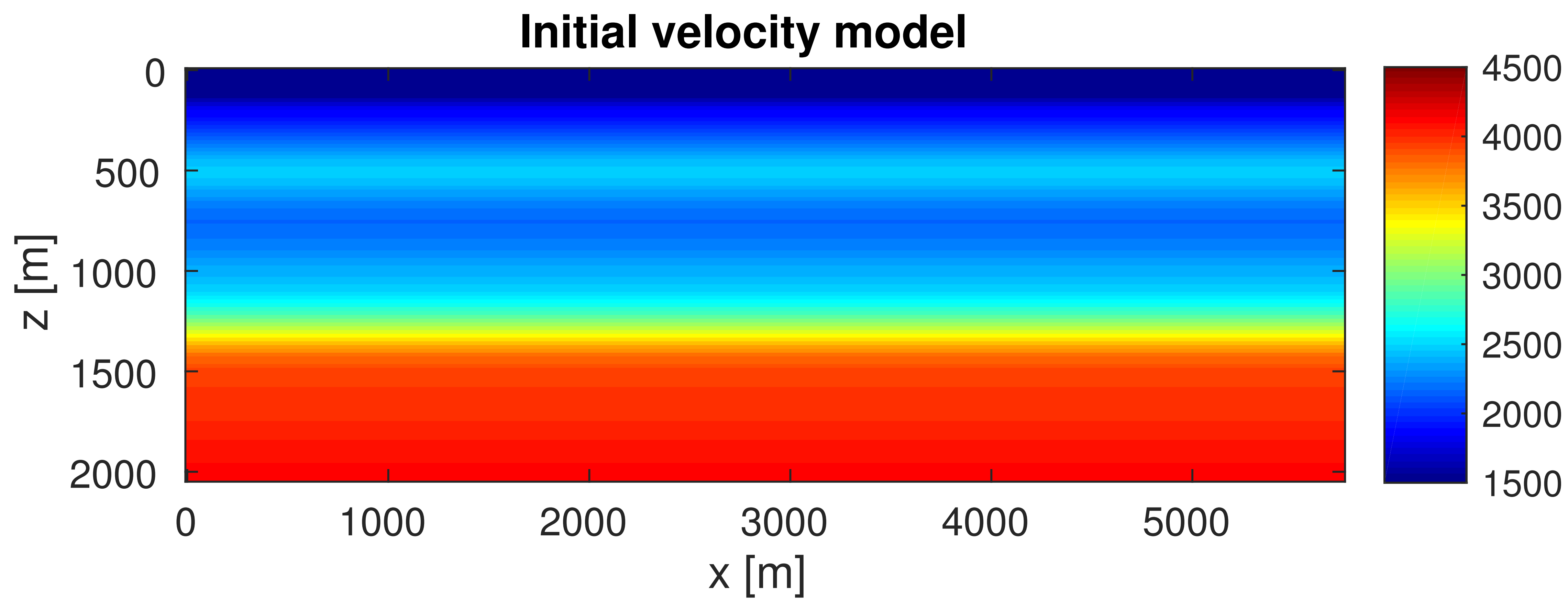
Convex & non-convex constraint sets for full-waveform inversion

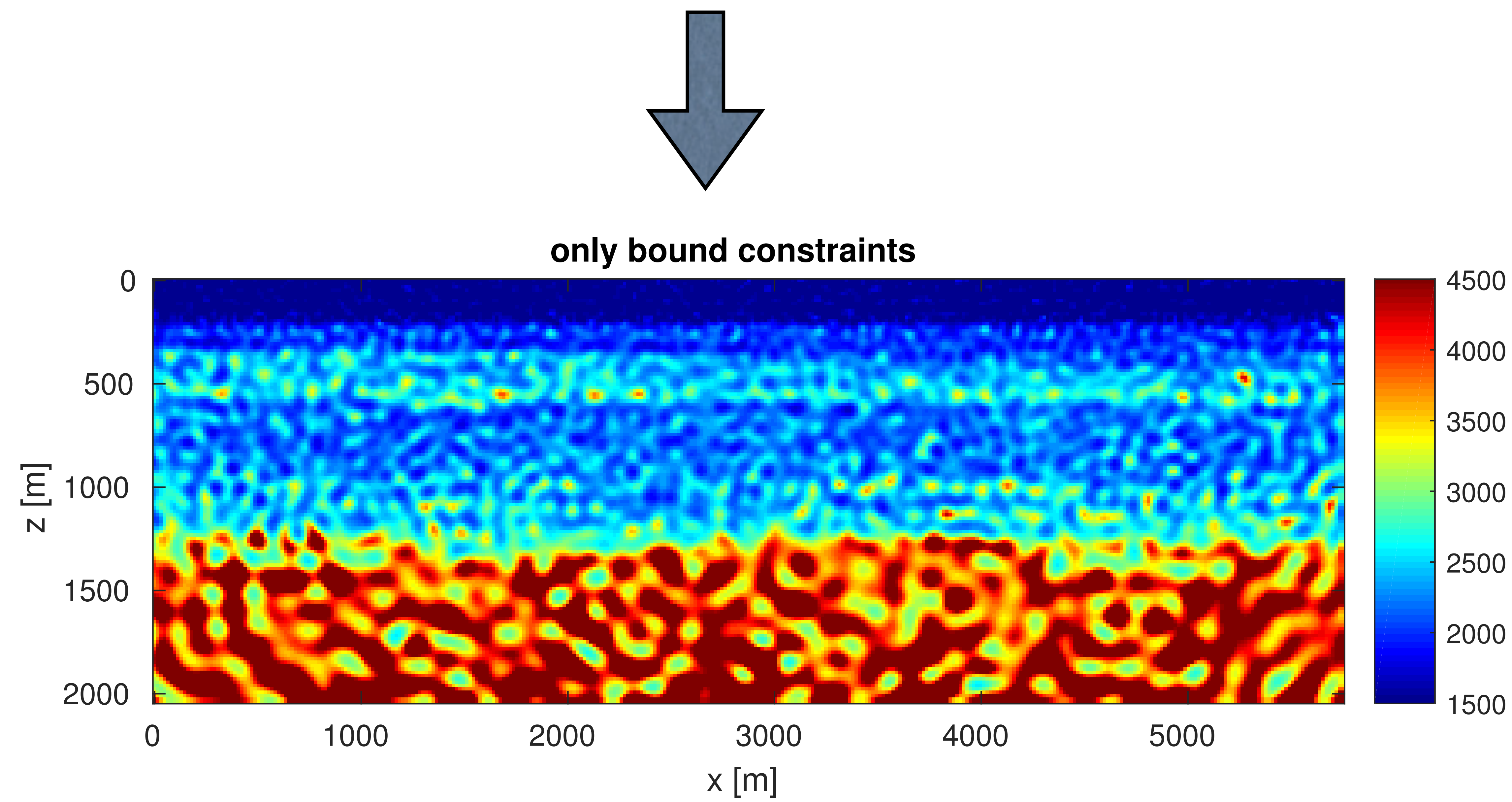
Bas Peters

SLIM
University of British Columbia

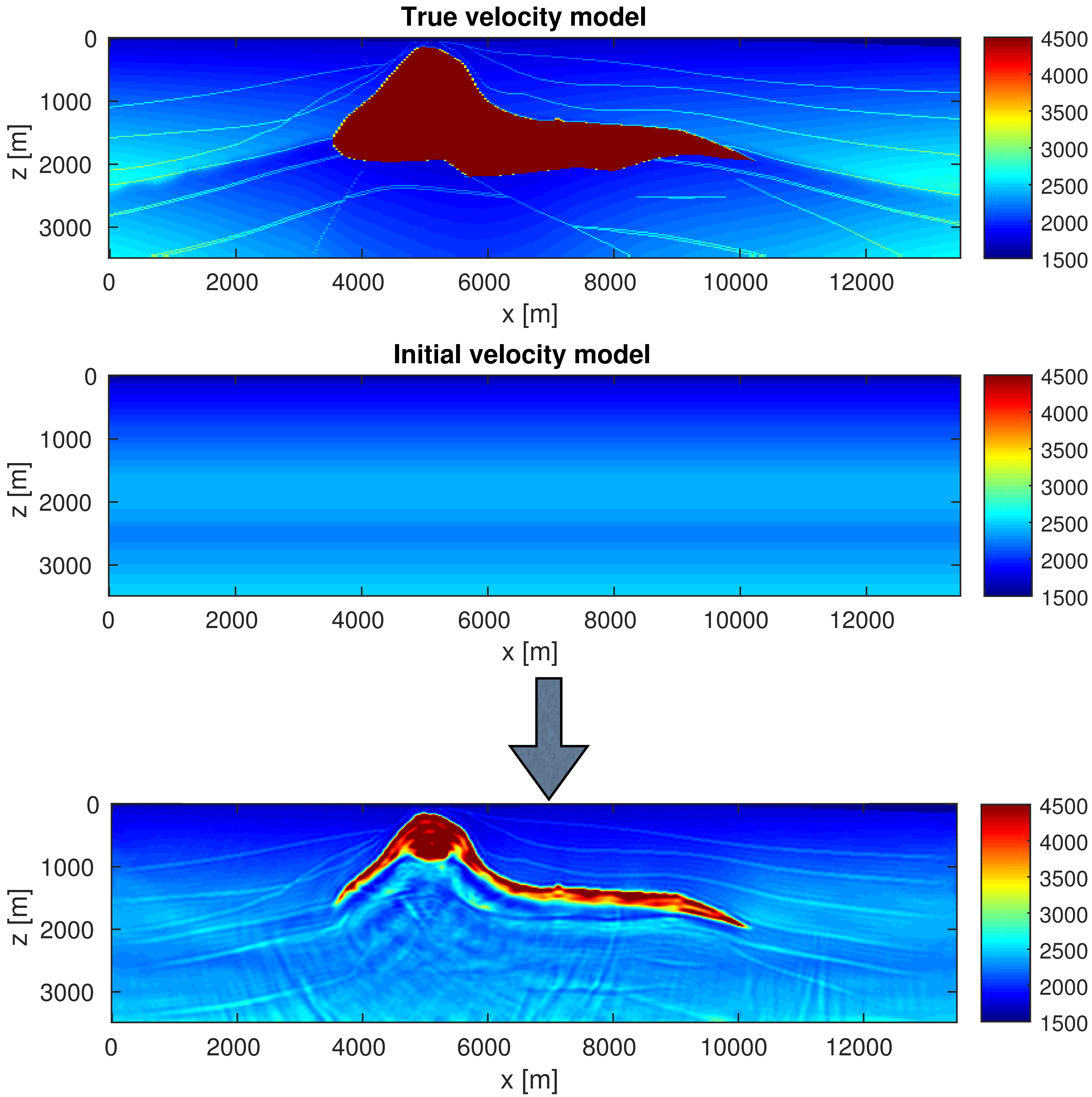
Motivation – noisy data



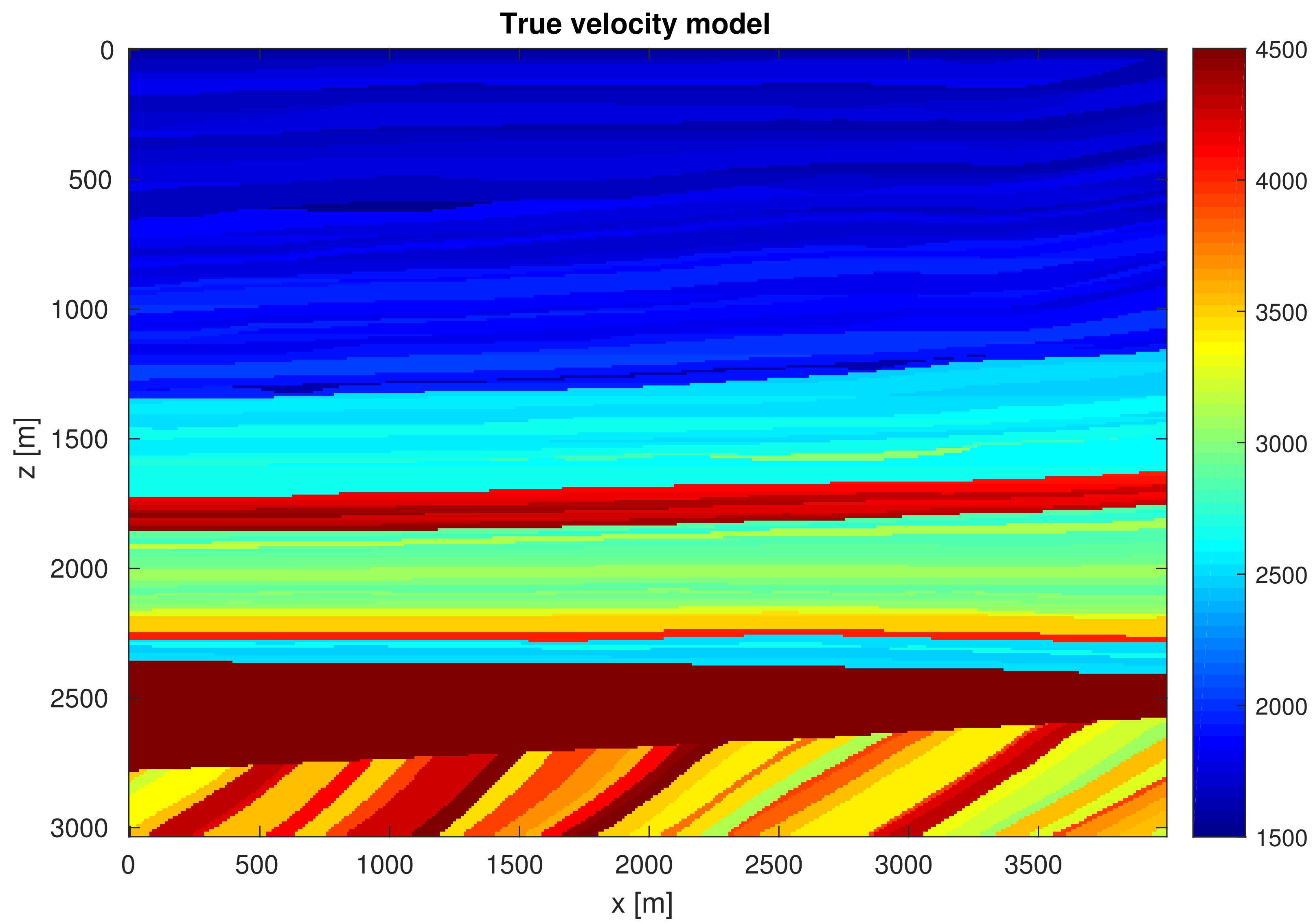


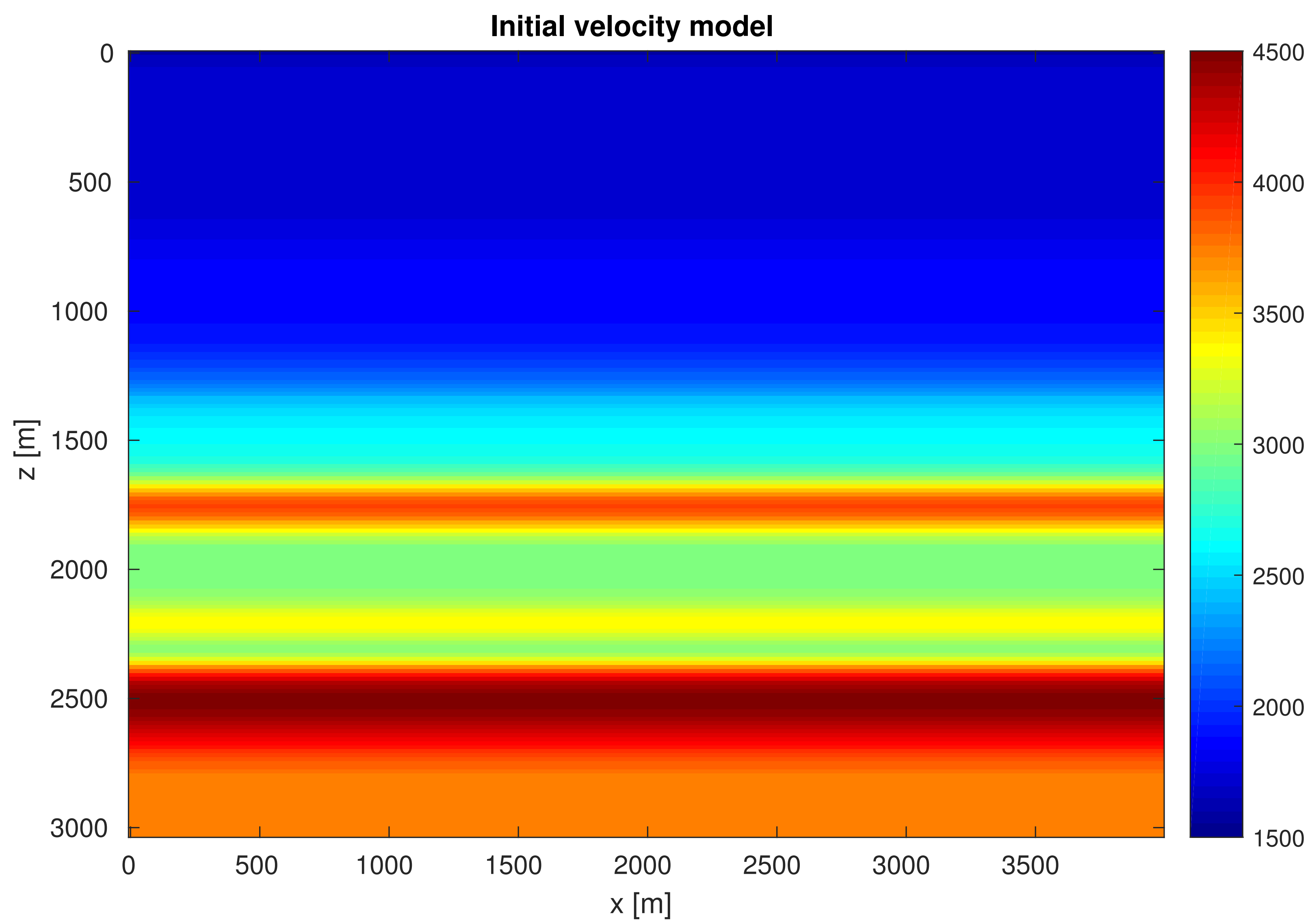


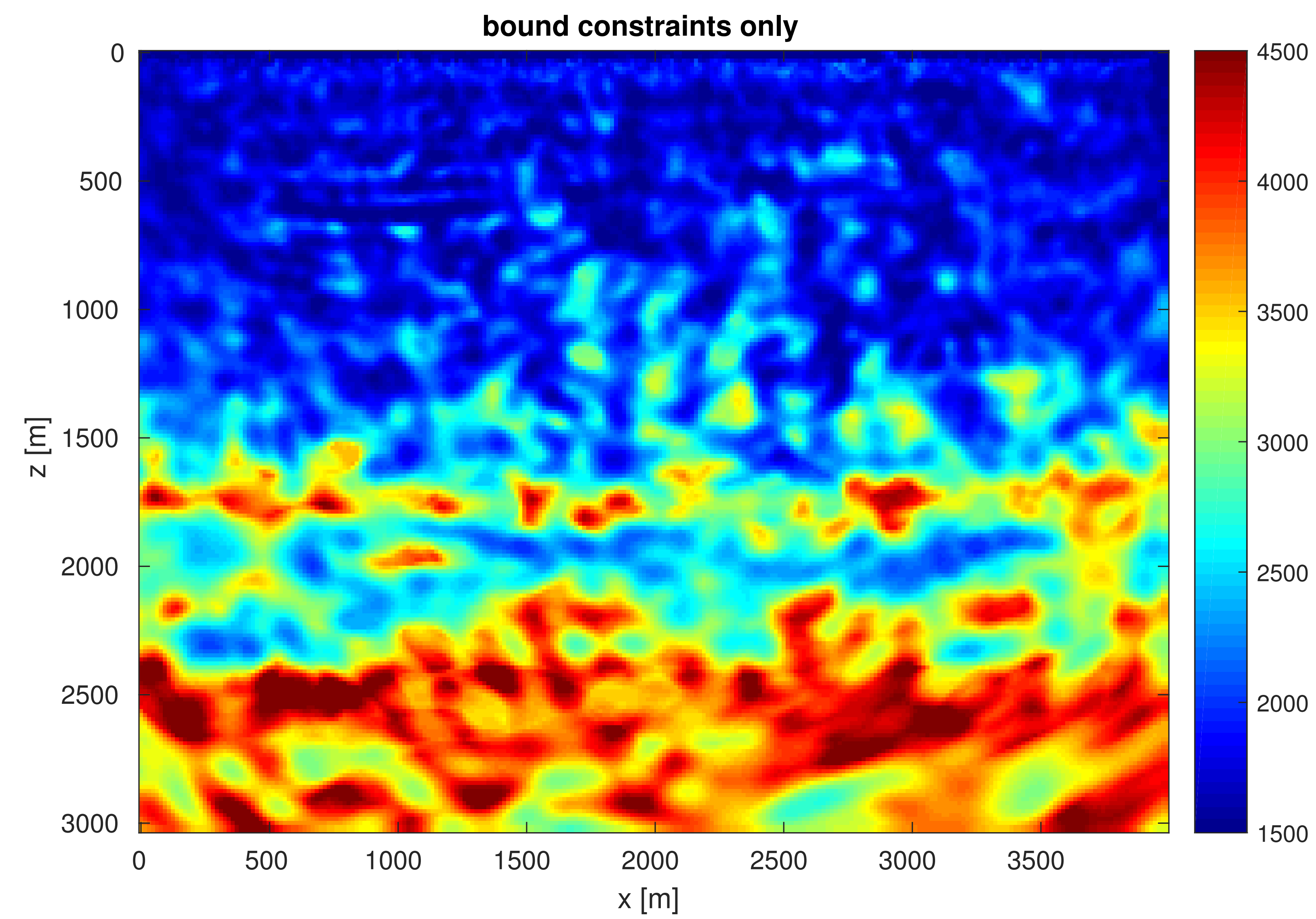
Motivation – bad start model / missing low freq.



Motivation – noisy data & few simultaneous sources







Motivation

Develop constraints & optimization to deal with these issues.

Constraints encode information about

- smoothness
- blockiness
- approximately layered media
- number of velocity jumps up or down
- maximum and minimum values, well-log information, reference models
- much more

Goal

Create software toolbox which builds on top of existing codes:

- use any code which provides function value and gradient
- applies to any non-linear inverse problem
- define arbitrary combinations of convex and non-convex constraint sets
- all iterates satisfy all constraints
- convenient translation of prior information into constraints
- data-misfit function and constraints are uncoupled

Constraints

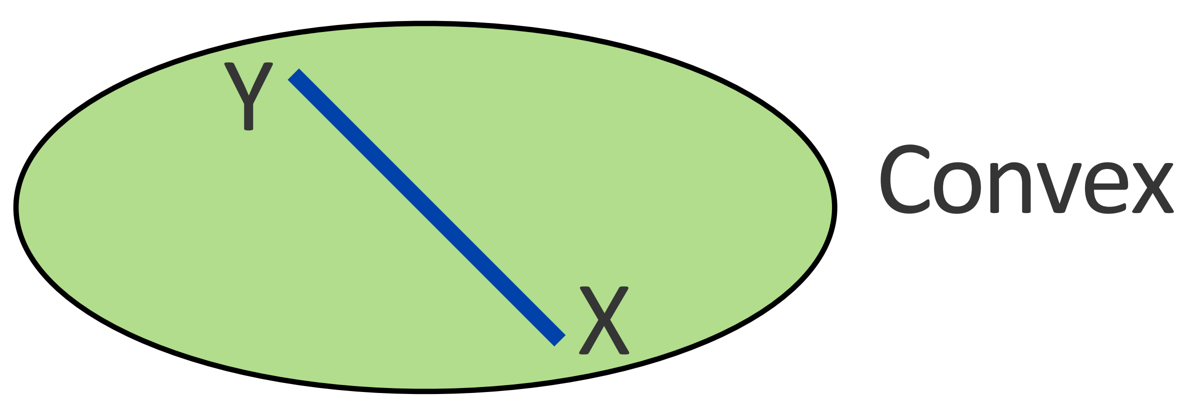
Currently implemented:

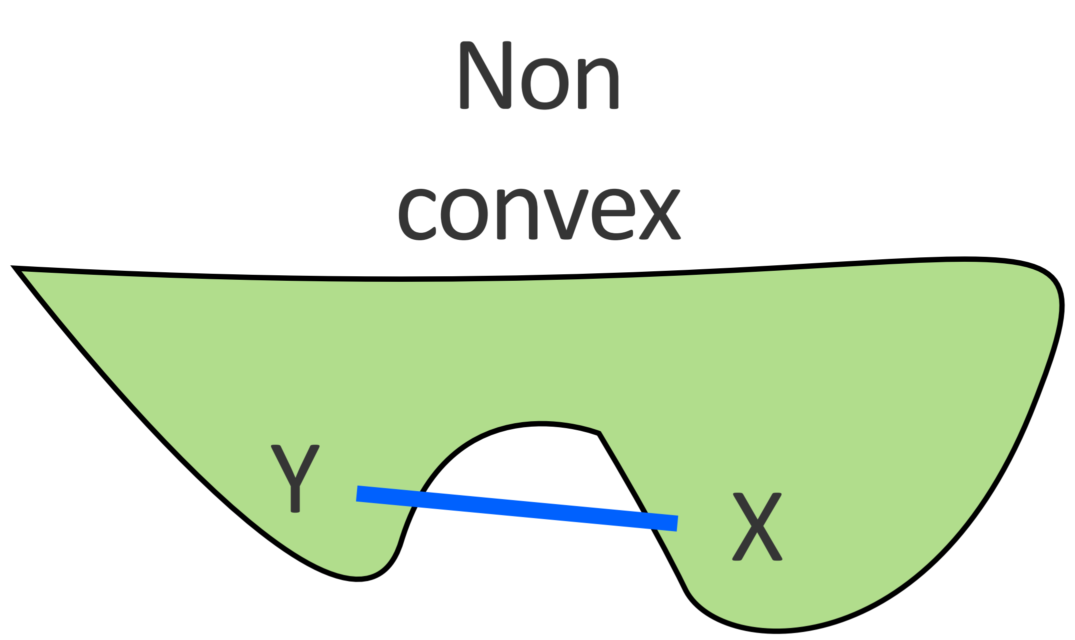
- bounds
- nuclear norm, rank
- ℓ_1 - based sparsity promotion total-variation/transform-domain sparsity
- cardinality (ℓ_0) - based total-variation transform-domain sparsity constraints
- slope constraints / transform-domain bounds
- Fourier-domain smoothness / subspace constraints

Convex sets : some properties

Convex set

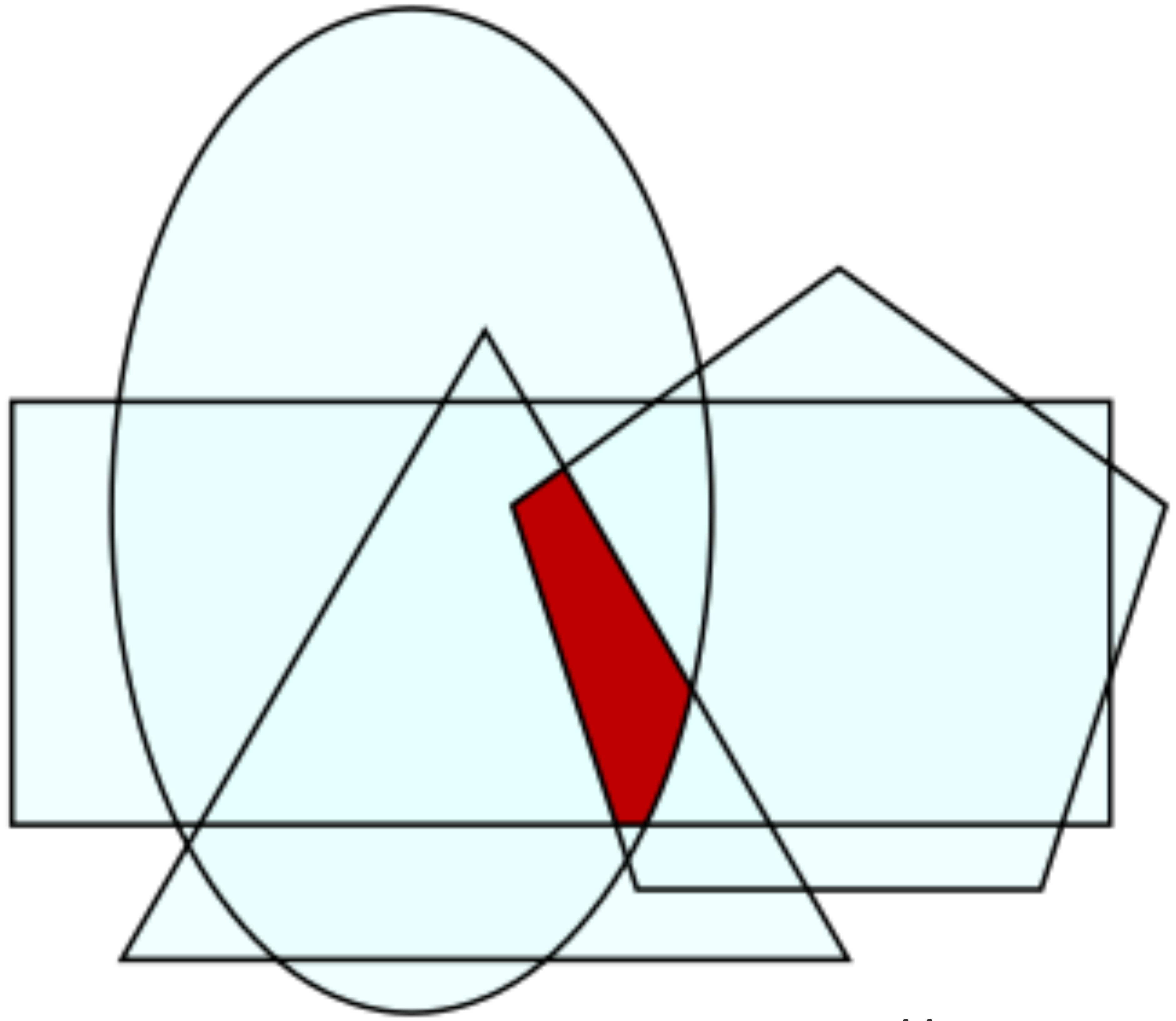
- there is a linear path contained in the set between every pair of the set
- every point is linearly reachable from another point
- projection onto a convex set is unique
- projection onto a convex set is a non-expansive operation





Convex sets : intersections

intersection of convex sets is also convex



https://en.wikipedia.org/wiki/Helly%27s_theorem

Prior information as convex sets

Projection (Euclidean, minimum-distance projection):

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{m}\|_2 \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}_1 \bigcap \mathcal{C}_2$$

Important property:

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \mathcal{P}_{\mathcal{C}}(\mathcal{P}_{\mathcal{C}}(\mathbf{m}))$$

From prior information to set definition

The next few slides show only a few examples.

Examples illustrate one set at a time.

In practice, we combine multiple sets. (shown later in this talk)

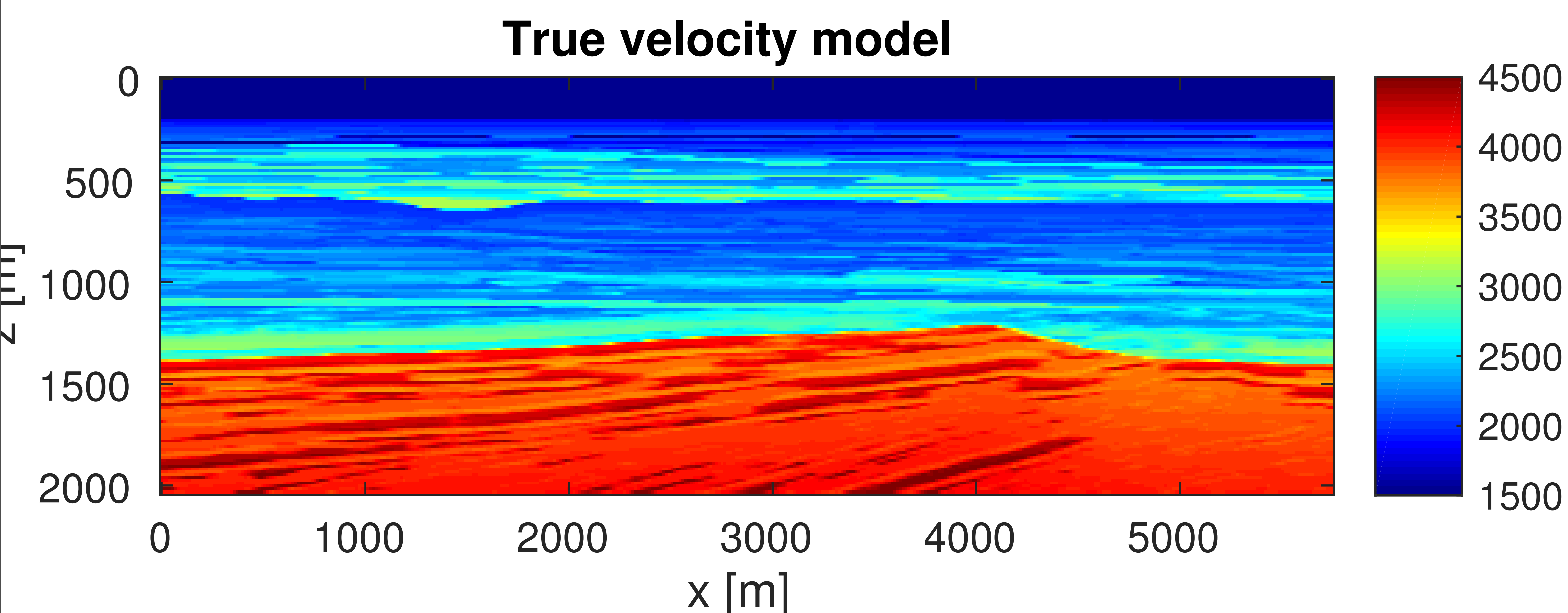
Note: constraints apply to a discrete image, not the true Earth properties.

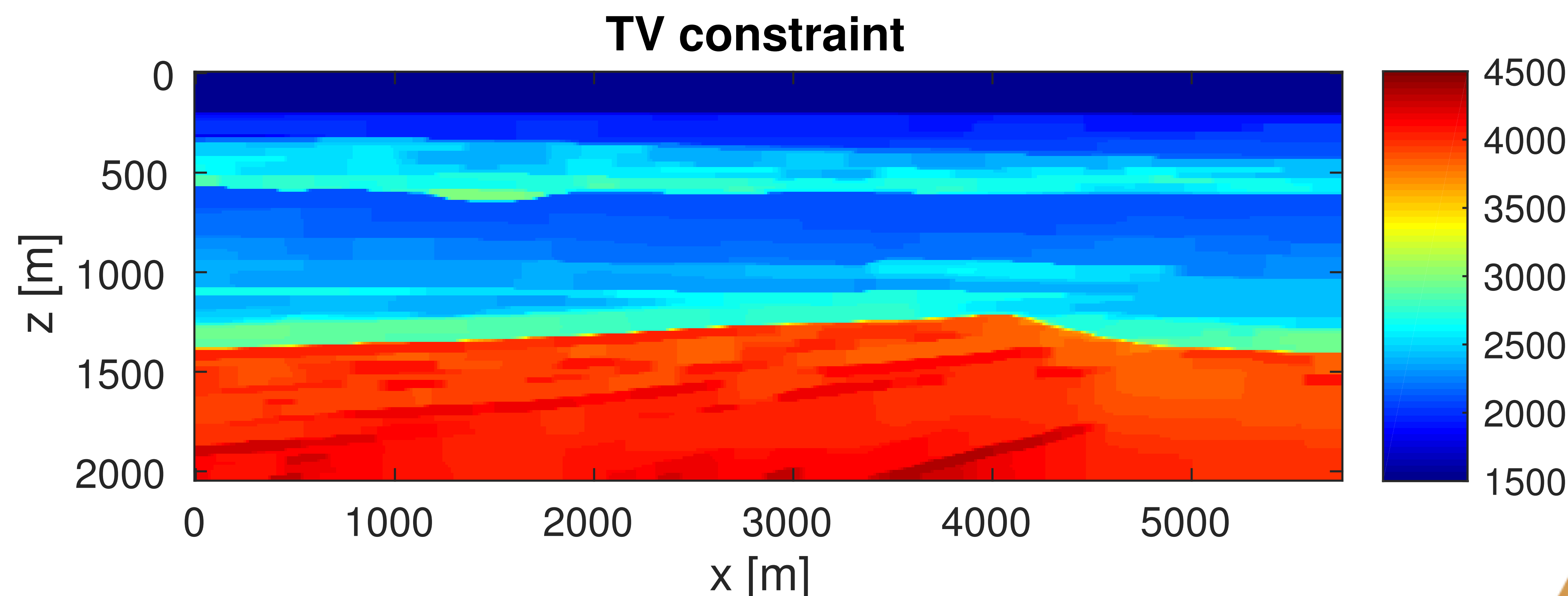
Convex transform-domain sparsity promotion

$$\mathcal{C} \equiv \{\mathbf{m} \mid \|A\mathbf{m}\|_1 \leq \sigma\}$$

Request a few significant nonzero coefficients in transform domain

Transform domain examples: Wavelet, Curvelet, TV, discrete gradient, ...



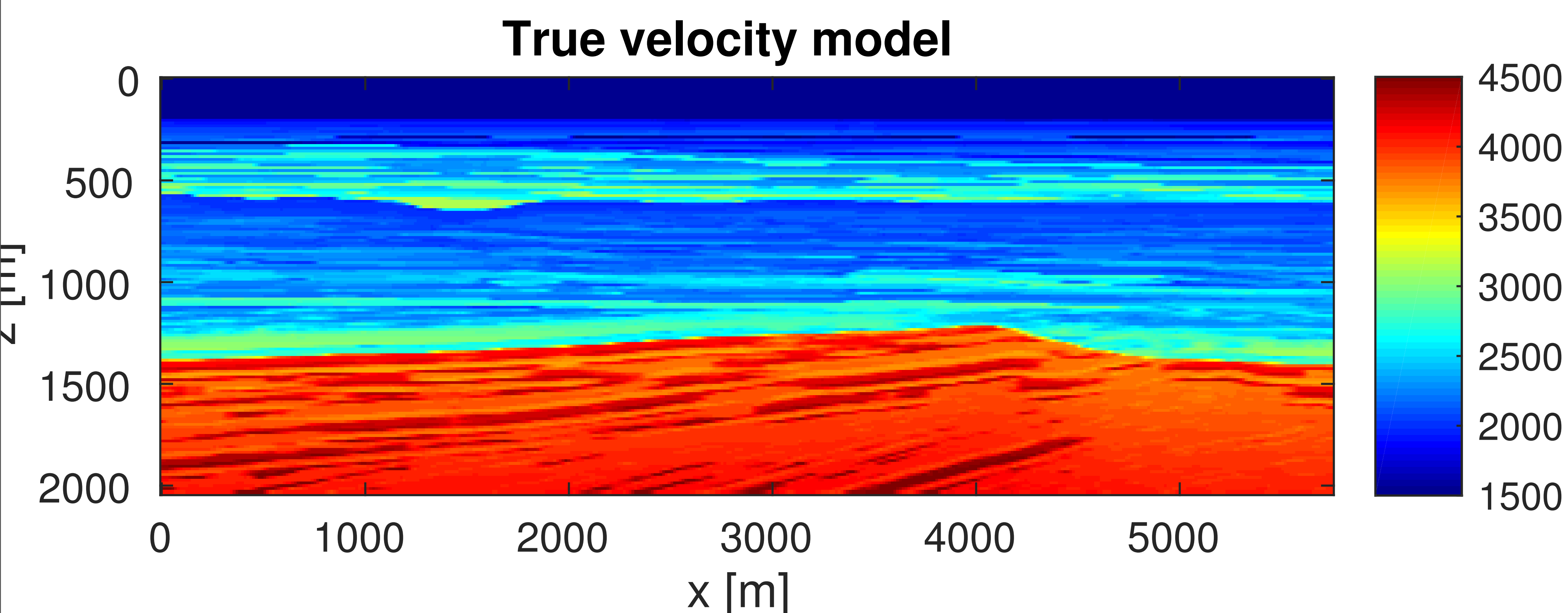


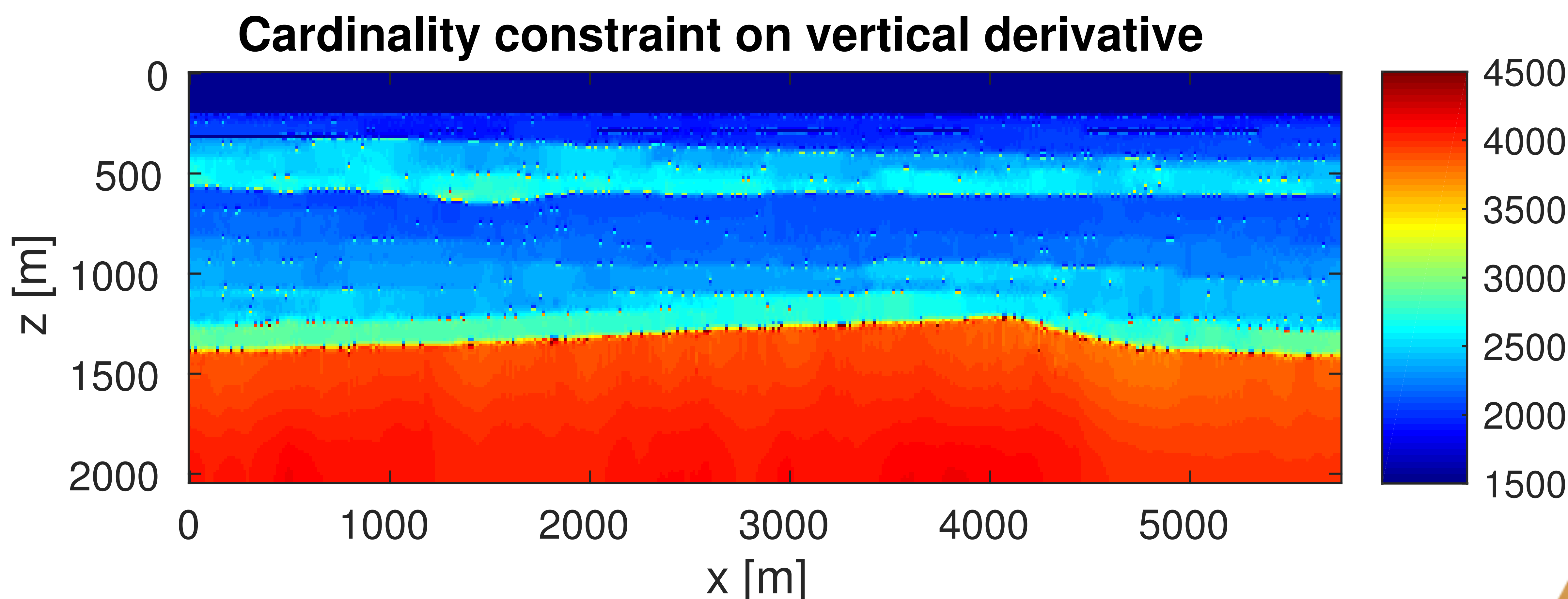
Non-convex transform-domain cardinality constraints

non-convex set: $\mathcal{S} \equiv \{\mathbf{m} \mid \mathbf{card}(A\mathbf{m}) \leq k\}.$ k : integer

Cardinality of discrete vertical derivative is set to: expected number major horizontal interfaces – 1 .

Artifacts are generally not a problem if used in combination with other constraints.





Non-convex transform-domain cardinality constraints

non-convex cardinality:

requires estimate of the **number** of major interfaces.

$$\mathcal{S} \equiv \{\mathbf{m} \mid \mathbf{card}(A\mathbf{m}) \leq k\}$$

convex 1-norm:

requires estimate of the **number** of major interfaces and the **magnitude** of the jumps

$$\mathcal{C} \equiv \{\mathbf{m} \mid \|A\mathbf{m}\|_1 \leq \sigma\}$$

Rank constraints

non-convex set: $\mathcal{S}_1 \equiv \{M_r \mid M_r = \sum_{j=1}^{r} \lambda_j \mathbf{u}_j \mathbf{v}_j^*\}$

Simplest form of the projector: SVD, r<k

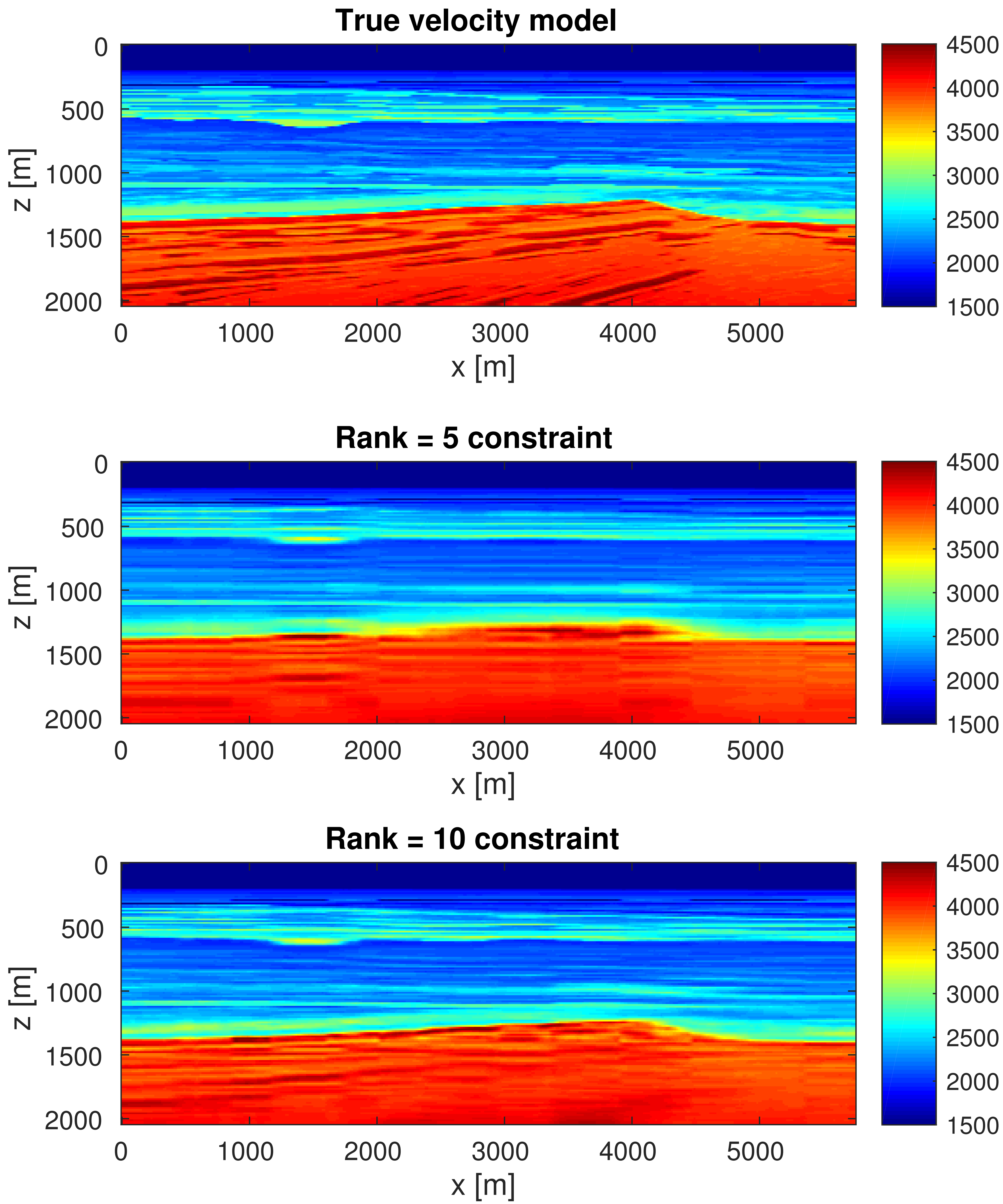
$$\mathcal{P}_{\mathcal{S}_1}(M) = \sum_{j=1}^{r} \lambda_j \mathbf{u}_j \mathbf{v}_j^*, \quad \text{with} \quad M = \sum_{j=1}^{k} \lambda_j \mathbf{u}_j \mathbf{v}_j^*.$$

Layered models are rank-1

Laterally invariant start models are rank-1

Rank constraints

projection of true model



Rank constraints

Approximately layered models are low rank, but not all low rank models are approximately layered.

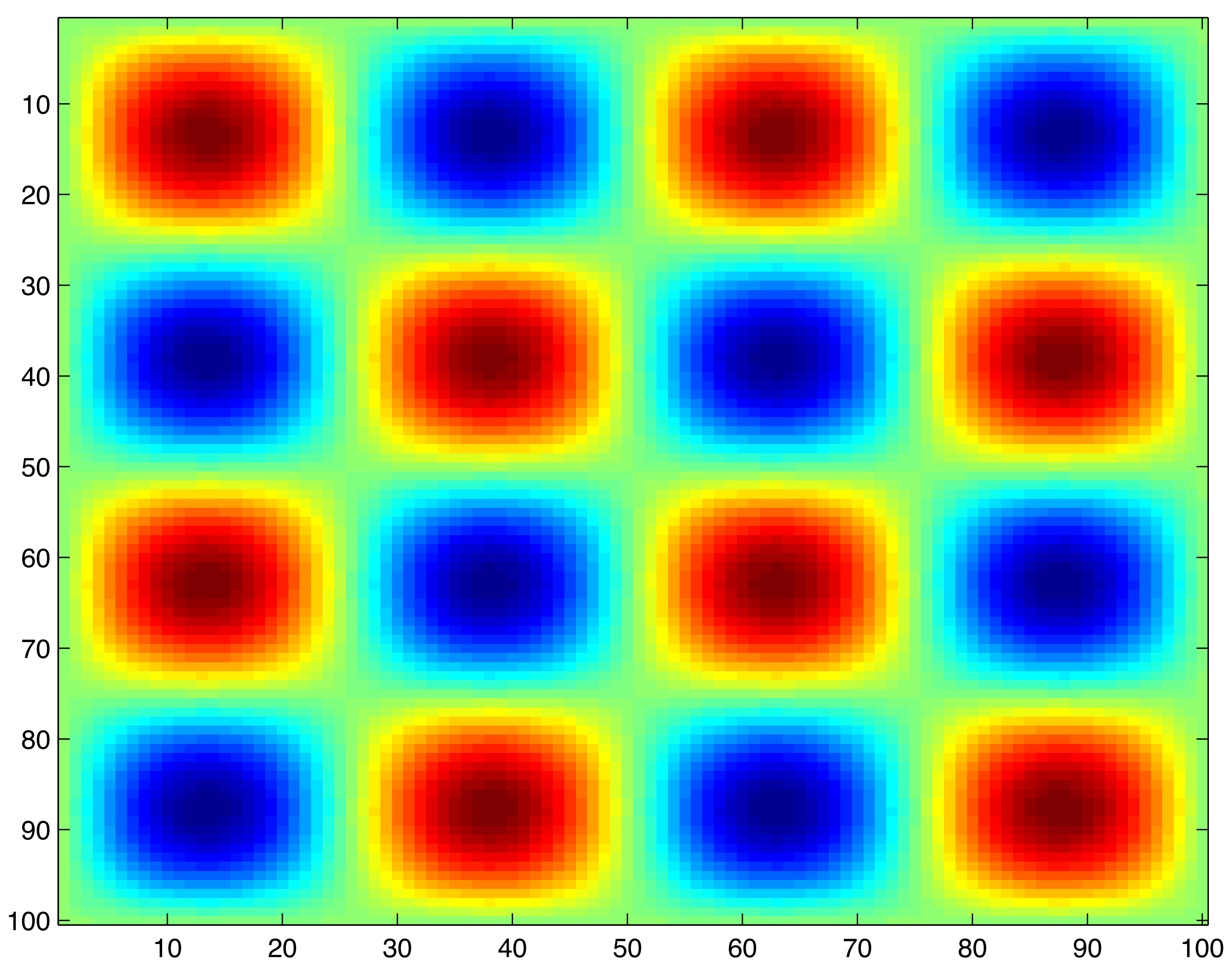
Rank describes a variety of ‘simple’ matrix structures.

Interesting feature:

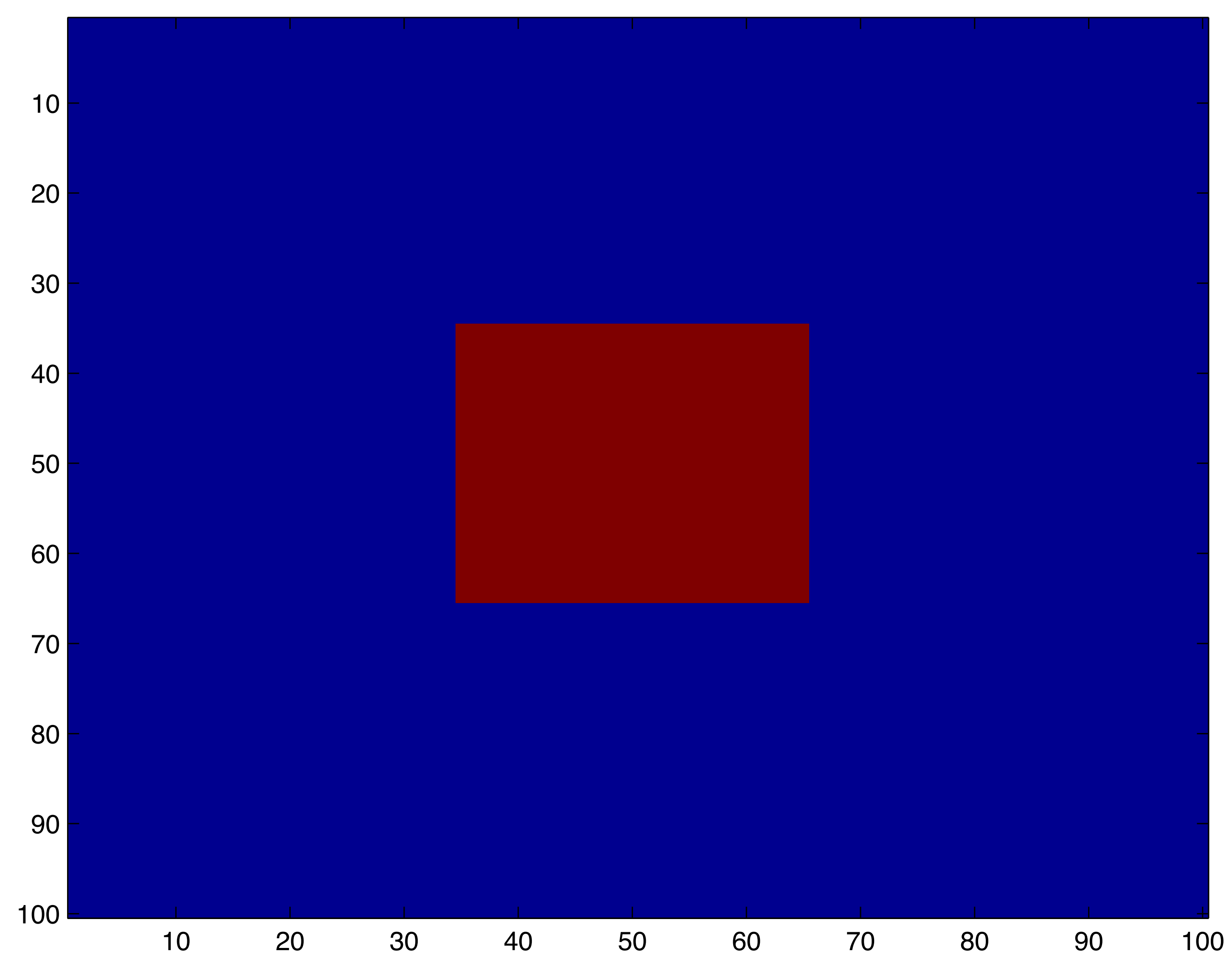
Media with smooth and blocky parts can be low-rank.

Rank-constraint

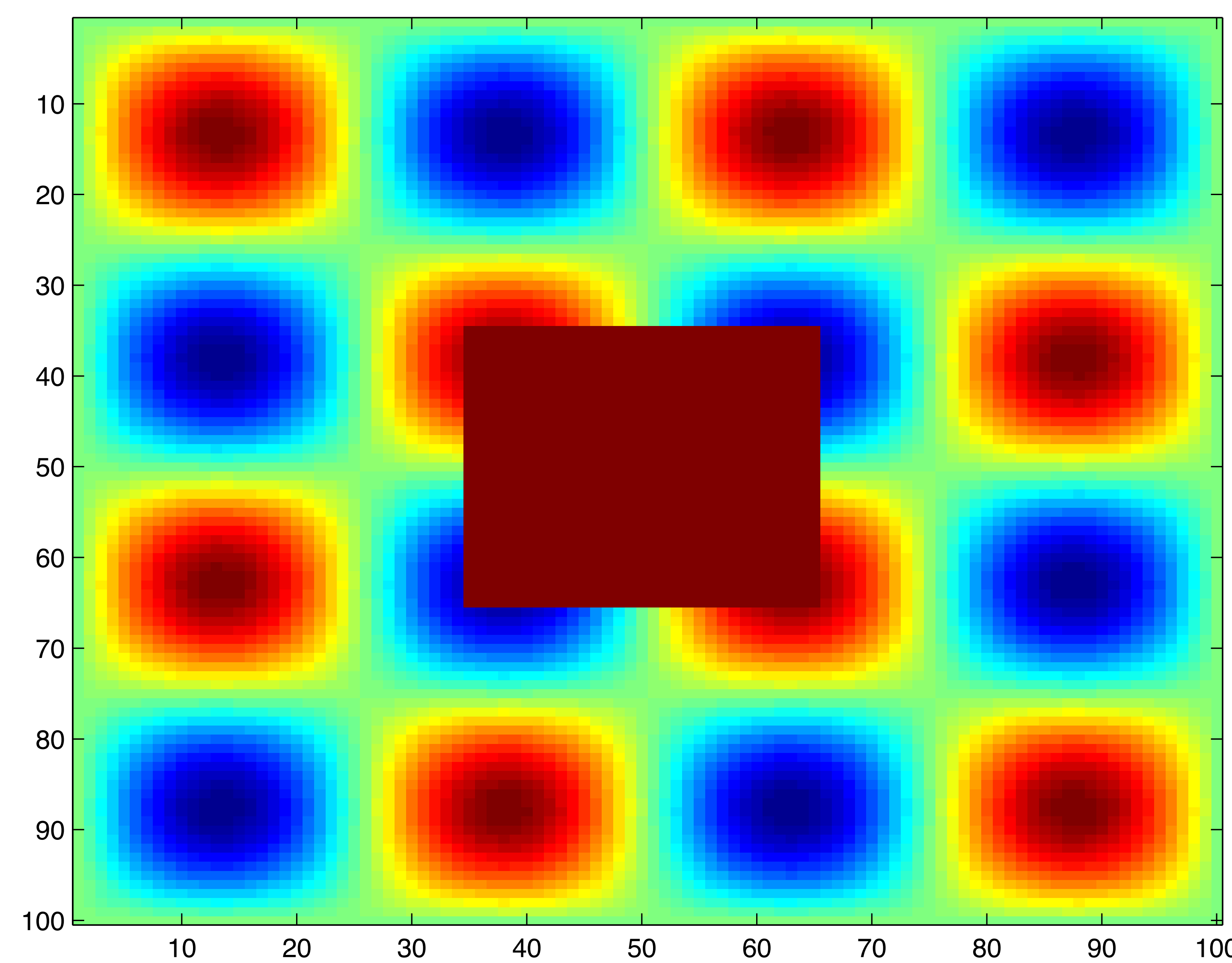
some very low rank examples:



Rank=1



Rank=2



numerical rank=3

Nuclear norm constraint

Nuclear norm:

- sum of singular values of a matrix
- heuristic for the rank
- less intuitive than rank, but a convex set
- personally, found it difficult to use

Transform-domain bound constraints / slope constraints

Element-wise bound-constraint on transform-domain coefficients:

$$\mathcal{C} \equiv \{\mathbf{m} \mid \mathbf{b}^l \leq A\mathbf{m} \leq \mathbf{b}^u\}$$

Not clear how this could help in Wavelet, Curvelet or Fourier-domain.

Useful in discrete-gradient domain:

$$A = I_n \otimes D_z \qquad D_z = \frac{1}{h_z}\begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

Transform-domain bounds / slope constraints

$$\mathcal{C} \equiv \{\mathbf{m}_i \mid \mathbf{b}_i^l \leq A\mathbf{m}_i \leq \mathbf{b}_i^u\} \text{ with } A = I_n \otimes D_z$$

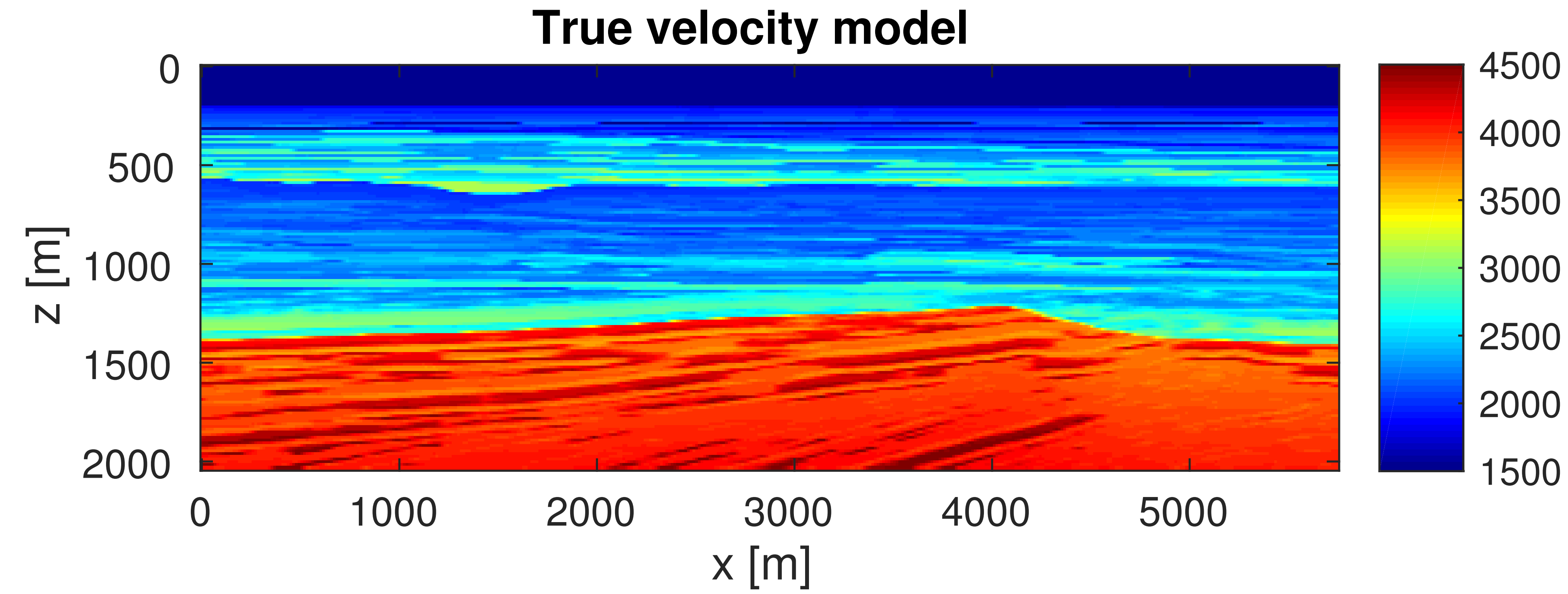
$$D_z = \frac{1}{h_z}\begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

Interpretation:

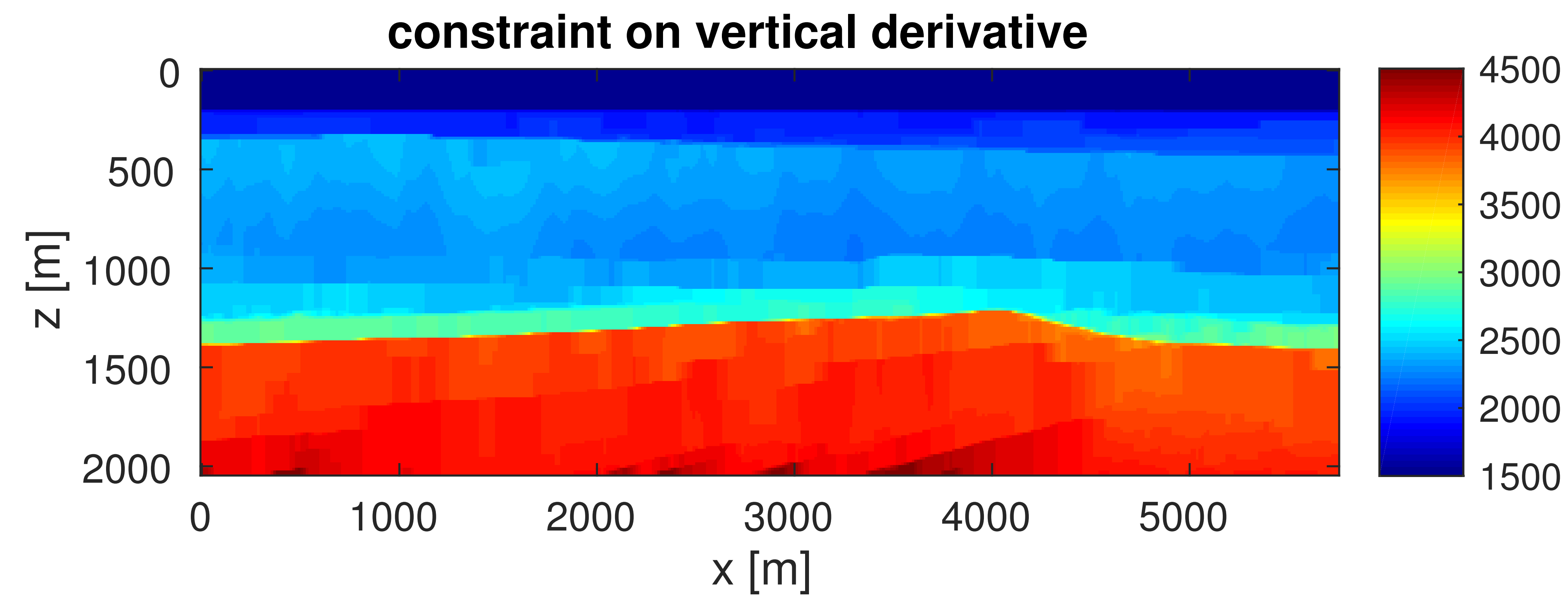
Limit the medium parameter variation per distance unit.

Can select different bounds for increasing values and decreasing values.

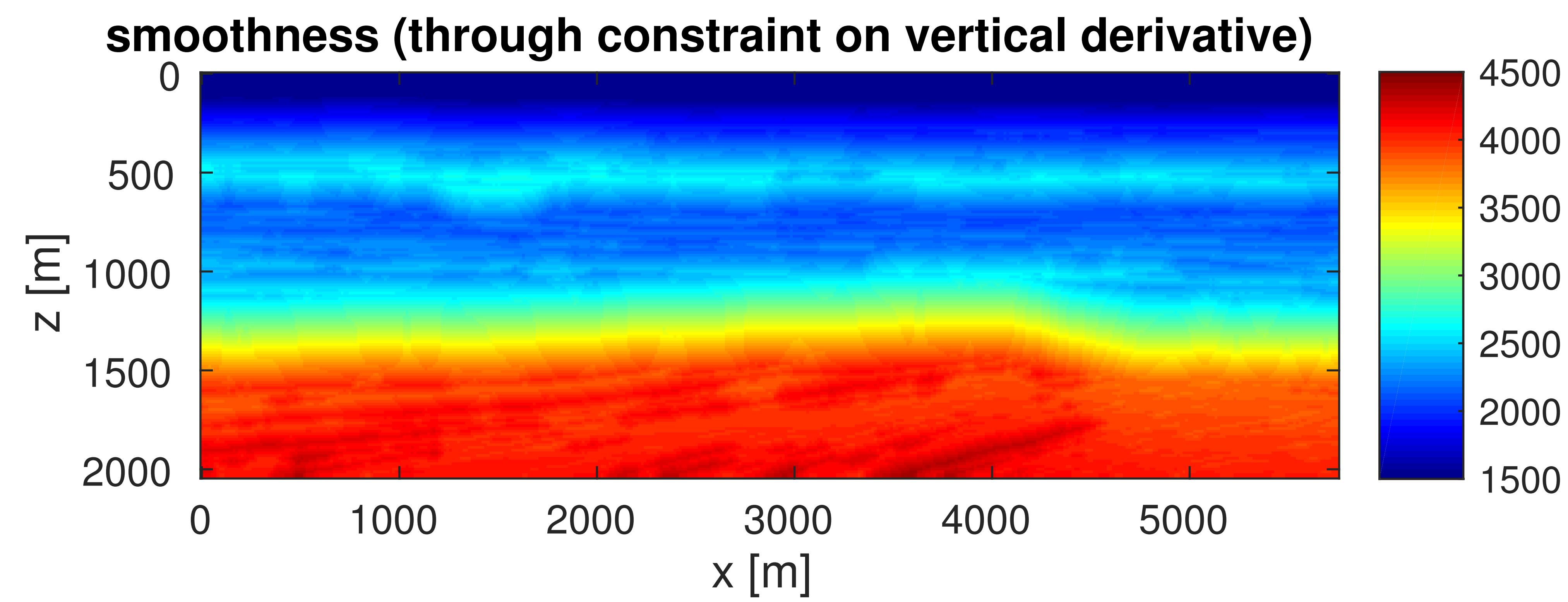
Transform-domain bound constraints



arbitrary medium parameter increase,
limited medium parameter decrease
with depth
->induces monotonicity



limited increase and limited decrease
->induces vertical smoothness
->still allows small velocity jumps



Design principles

Constrained optimization: $\min_{\mathbf{m}} f(\mathbf{m}) \quad \text{s.t.} \quad \mathbf{m} \in \bigcap_{i=1}^{p} \mathcal{C}_i$

Software is designed to build on top of existing algorithms:

- use any code which provides function value and gradient
- need to provide projector onto each constraint set
- define arbitrary number of convex and non-convex constraint sets
- assumes nonempty intersection of constraints
- all iterates satisfy all constraints

SLIM

Nested optimization strategy

Solution is computed by 3 levels of nested optimization/computations:

1. Algorithm for nonconvex (smooth + nonsmooth) optimization

$$\min_{\mathbf{m}} f(\mathbf{m}) \quad \text{s.t.} \quad \mathbf{m} \in \bigcap_{i=1}^{p} \mathcal{C}_i$$

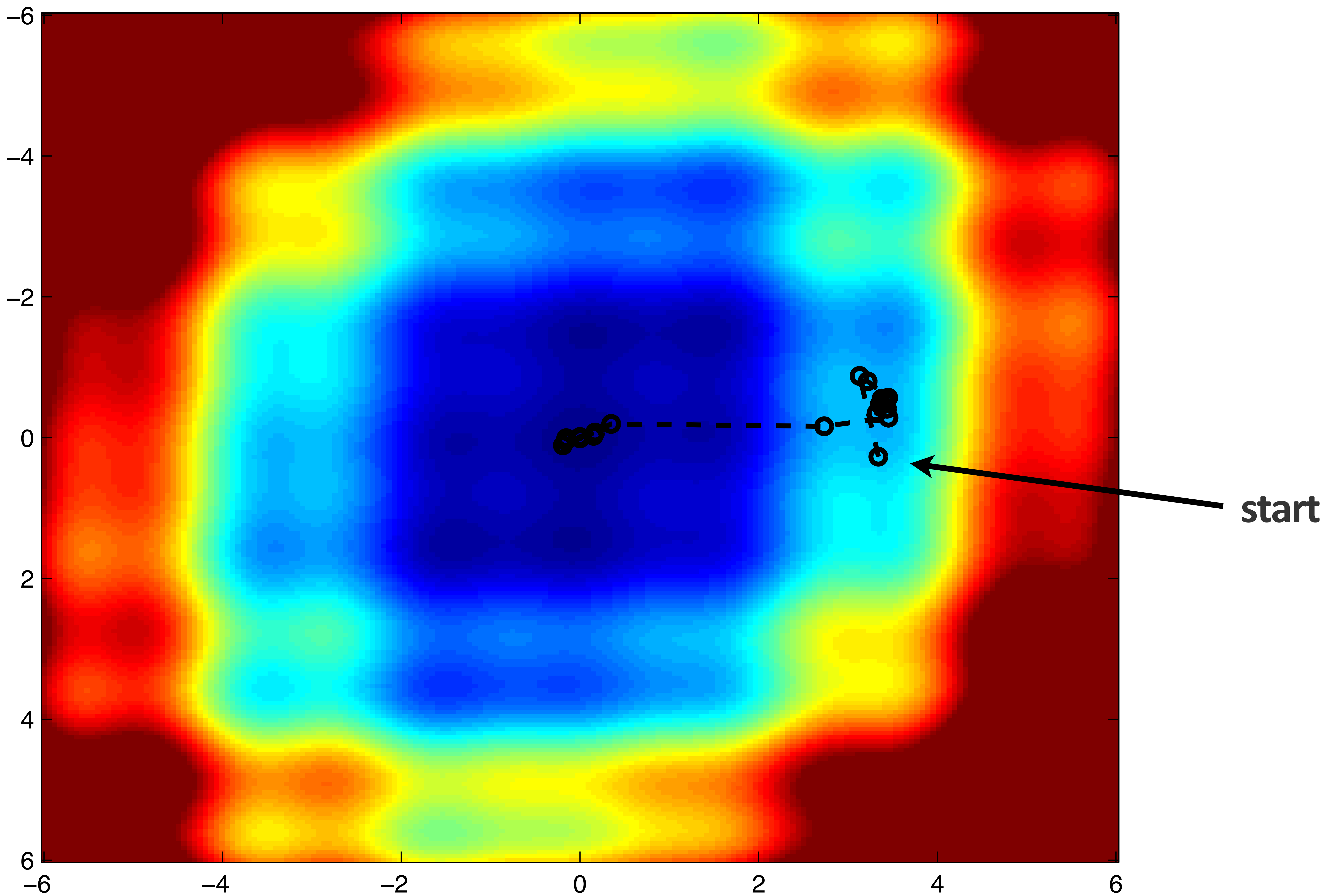
2. Algorithm computing the projection onto an intersection

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{m}\|_2 \quad \text{s.t.} \quad \mathbf{x} \in \bigcap_{i=1}^{p} \mathcal{C}_i.$$

3. Projection onto each set separately

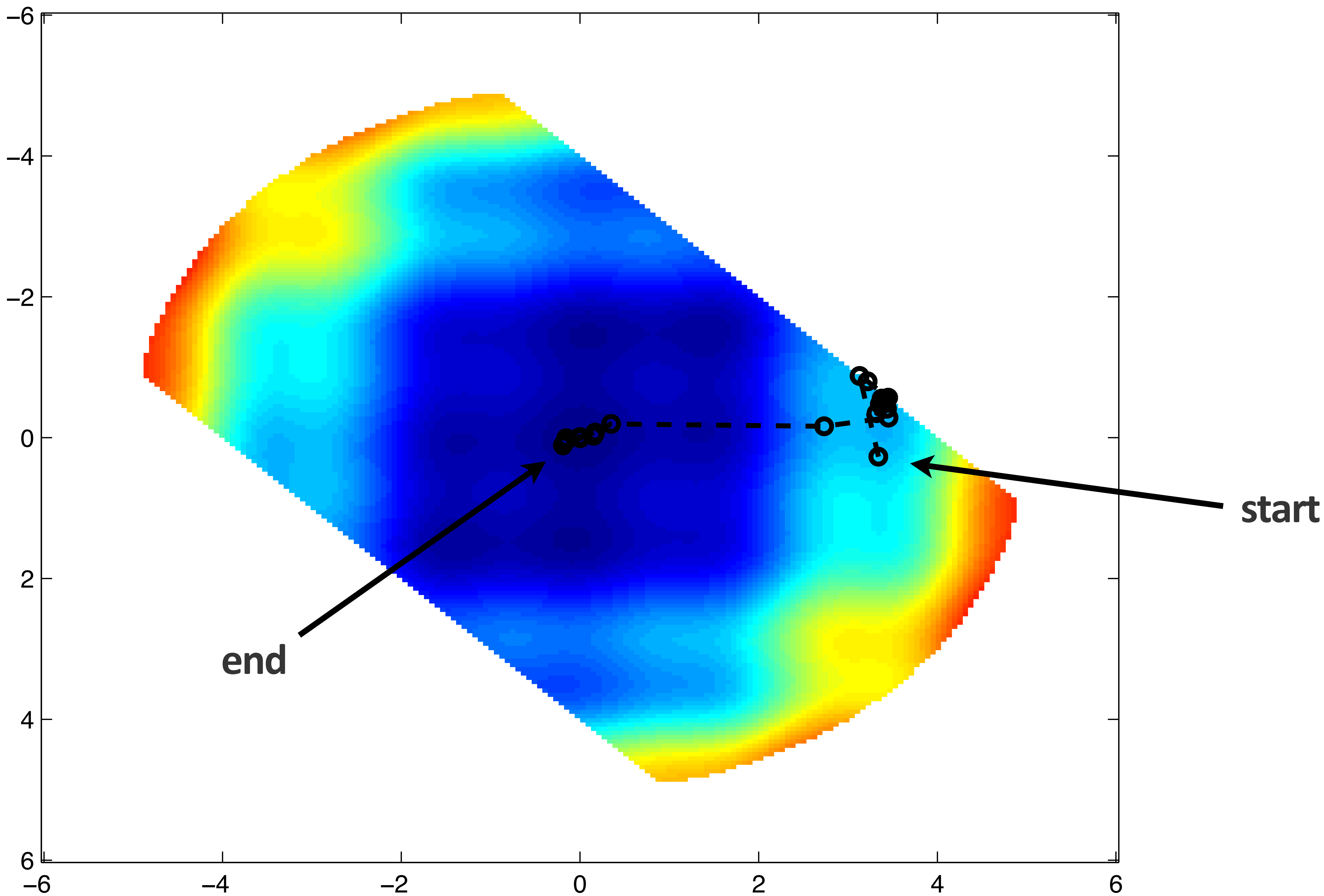
$$\mathcal{P}_{\mathcal{C}_i}(\mathbf{m}) = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{m}\|_2 \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}_i.$$

Projected gradient



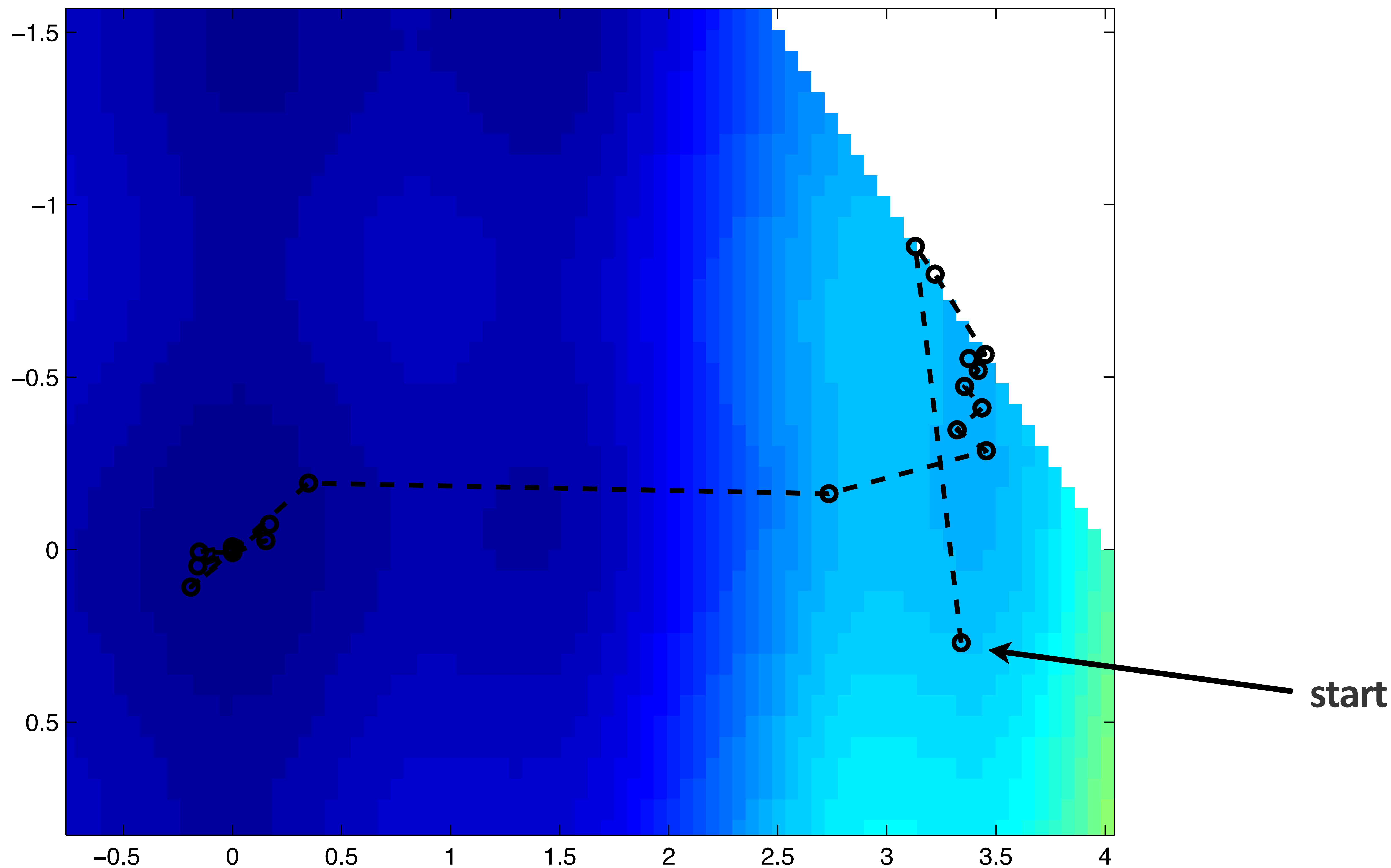
full non-convex function

Projected gradient



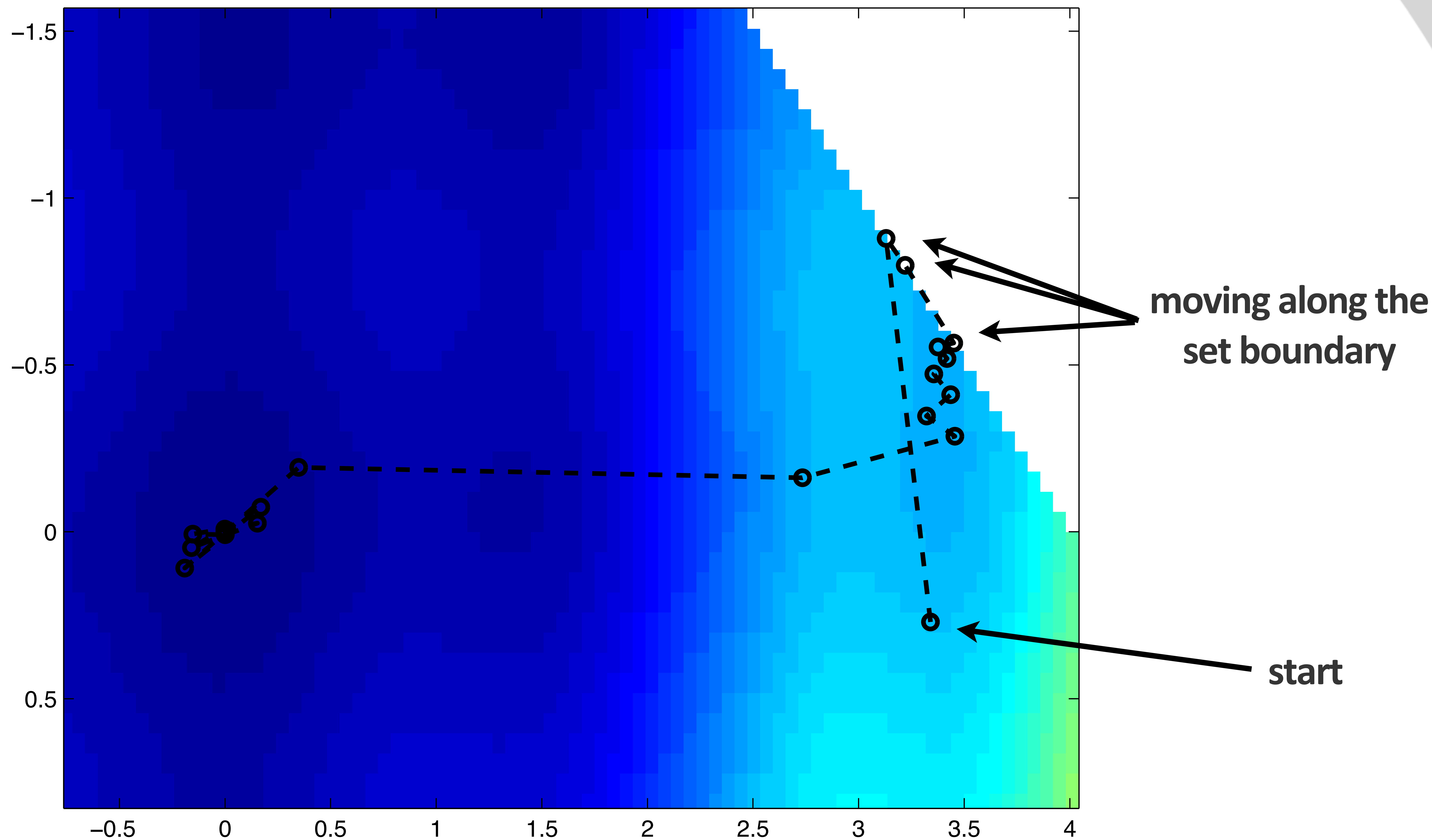
effective domain non-convex function

Projected gradient



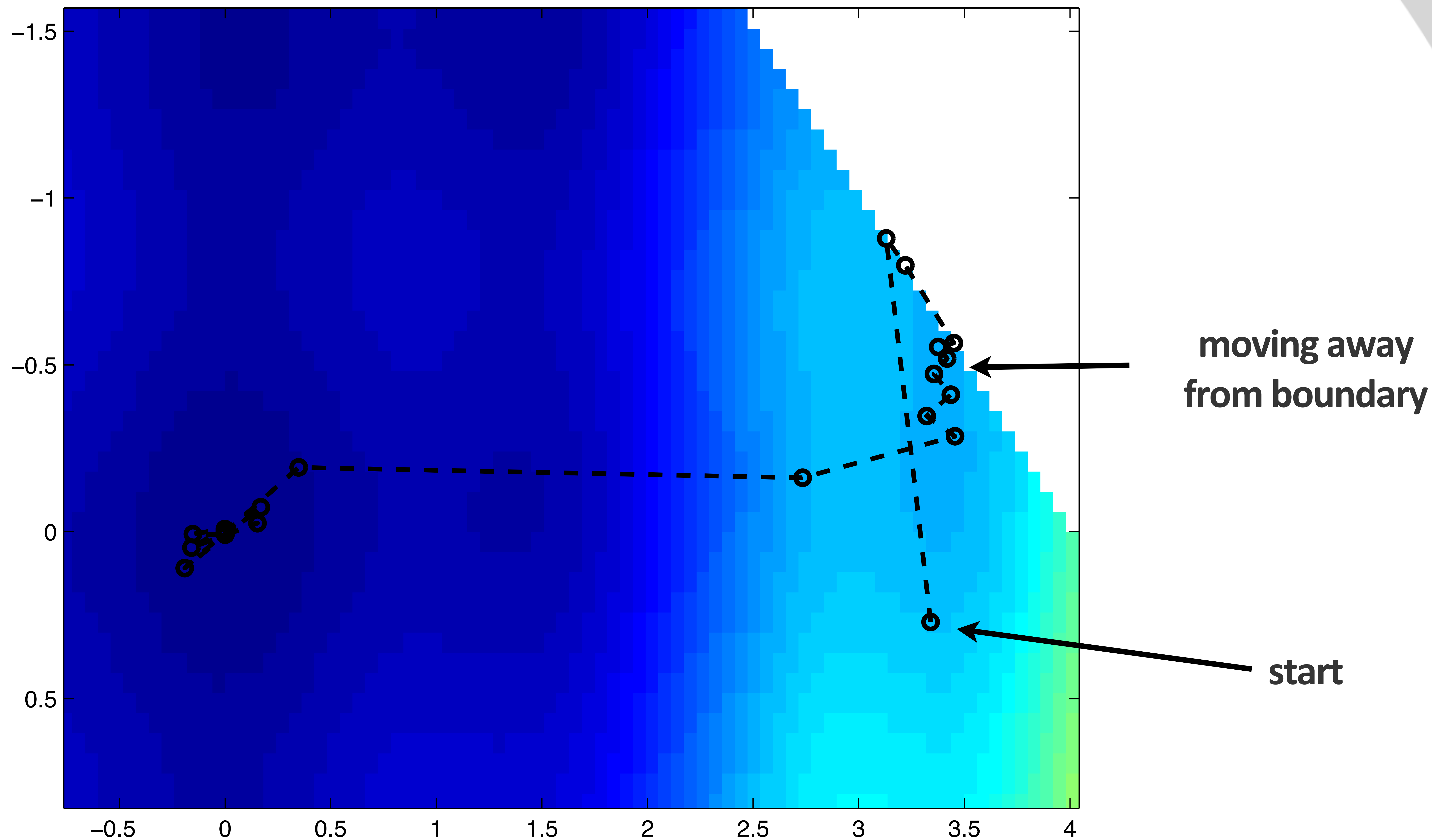
effective domain non-convex function
zoomed in

Projected gradient



effective domain non-convex function zoomed in

Projected gradient



effective domain non-convex function zoomed in

Dykstra's algorithm

Toy example:

find projection onto intersection of circle & square

Algorithm 1 Dykstra.

$x_0 = \mathbf{m},\ p_0 = 0,\ q_0 = 0$

For $k = 0, 1, \ldots$

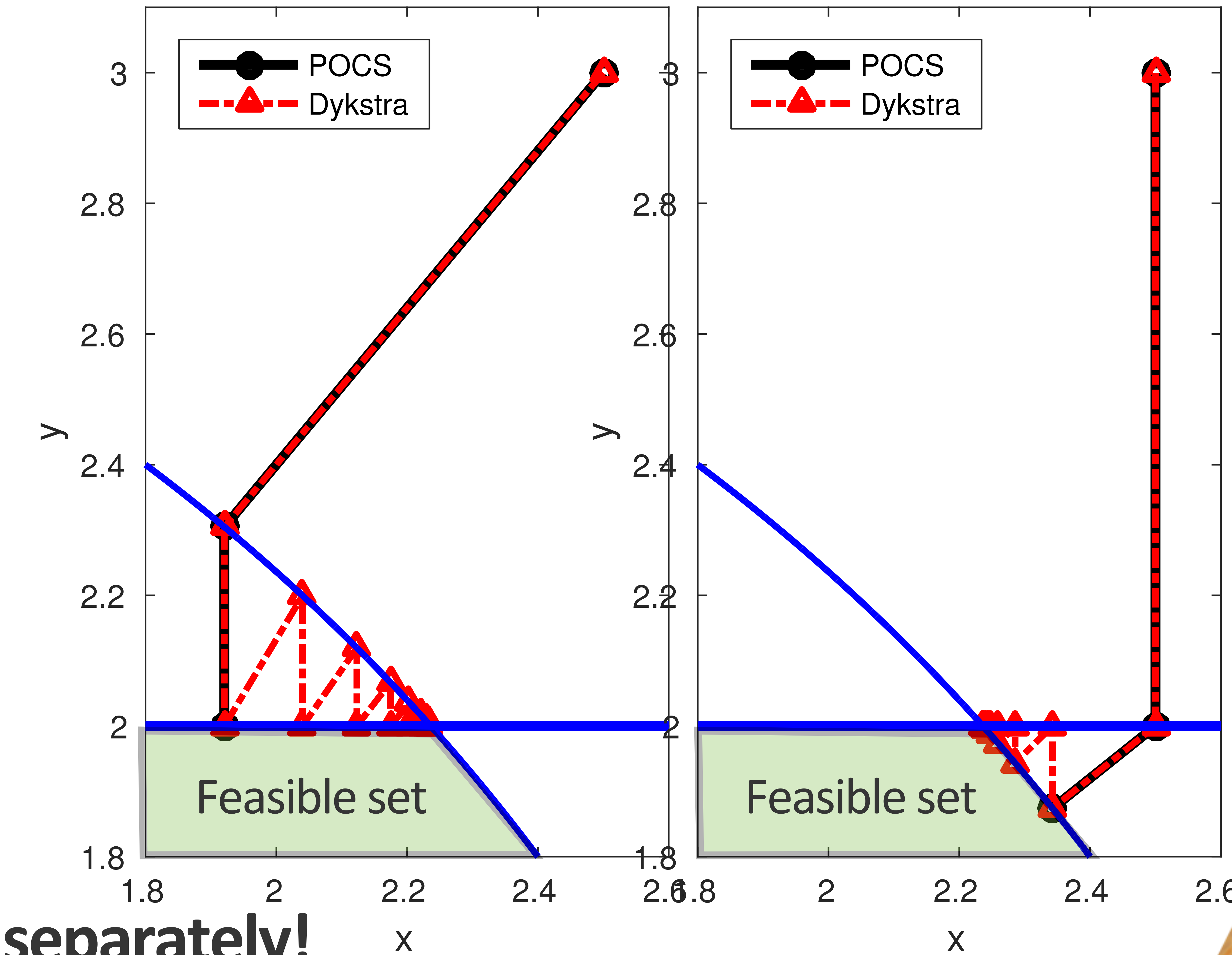
$\longrightarrow$ $\boxed{y_k = \mathscr{P}_{C_1}(x_k + p_k)}$

$p_{k+1} = x_k + p_k - y_k$

$\longrightarrow$ $\boxed{x_{k+1} = \mathscr{P}_{C_2}(y_k + q_k)}$

$q_{k+1} = y_k + q_k - x_{k+1}$

End



Only needs projections onto each set separately!

Projection computation

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C}$$

If no closed form solution is available: reformulate and solve using ADMM.

This works for all transform-domain norm/cardinality/bounds

$$\begin{aligned}
\mathcal{P}_{\mathcal{C}}(\mathbf{m}) &= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C} \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \|A\mathbf{x}\| \leq \sigma \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \|\mathbf{z}\| \leq \sigma\,,\ A\mathbf{x} = \mathbf{z}
\end{aligned}$$

SLIM

Projection computation

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C}$$

If no closed form solution is available: reformulate and solve using ADMM.

This works for all transform-domain norm/cardinality/bounds

$$\begin{aligned}\mathcal{P}_{\mathcal{C}}(\mathbf{m}) &= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C} \\ &= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{card}(A\mathbf{x}) \leq \sigma \\ &= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{card}(\mathbf{z}) \leq \sigma\,,\ A\mathbf{x} = \mathbf{z}\end{aligned}$$

Projection computation

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C}$$

If no closed form solution is available: reformulate and solve using ADMM.

This works for all transform-domain norm/cardinality/bounds

$$\begin{aligned}
\mathcal{P}_{\mathcal{C}}(\mathbf{m}) &= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{x} \in \mathcal{C} \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{b}^l \leq A\mathbf{x} \leq \mathbf{b}^u \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \mathbf{b}^l \leq \mathbf{z} \leq \mathbf{b}^u\,,\ A\mathbf{x} = \mathbf{z}
\end{aligned}$$

Projection computation

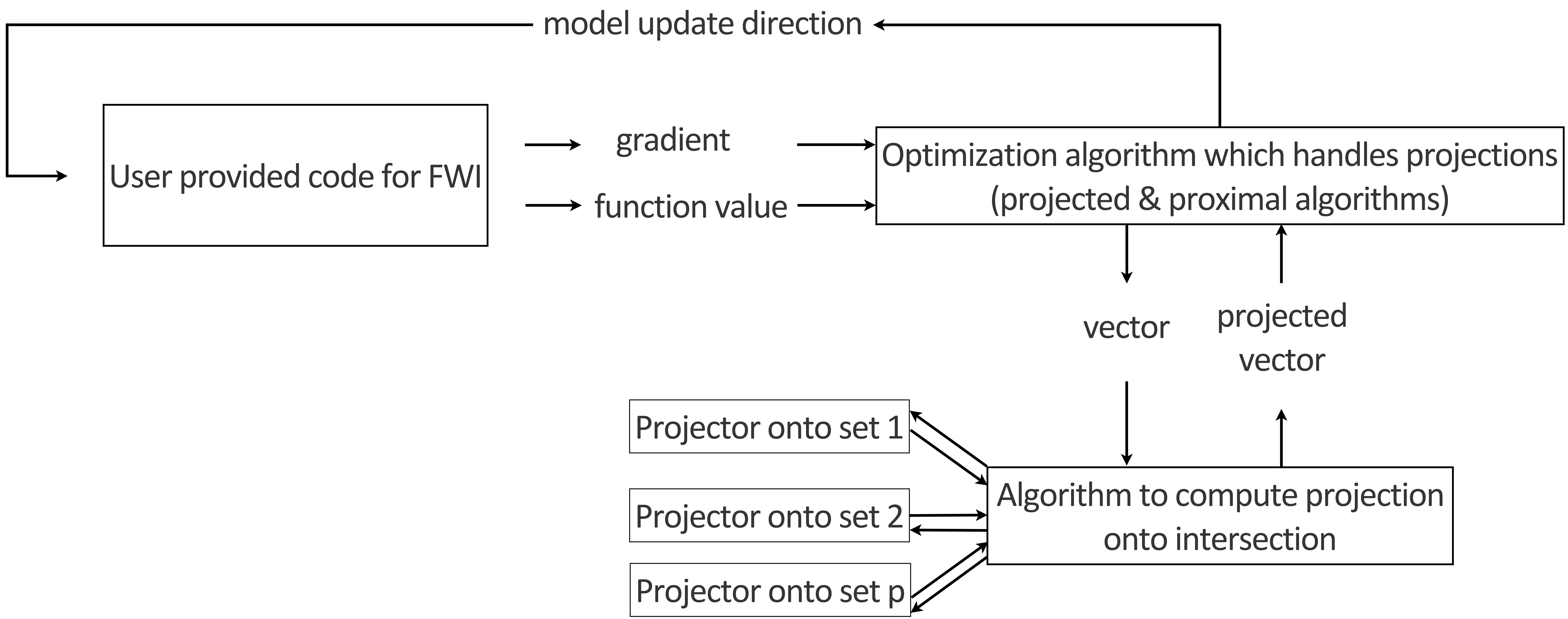
$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 \quad \text{s.t} \quad \|\mathbf{z}\| \leq \sigma \, , \; A\mathbf{x} = \mathbf{z}$$

```
%obtain projector function handle
%A: transform domain operator

%proj_z projector onto norm-ball, bounds or cardinality
proj_z=@(input) project_l1_norm(input,sigma)

proj_TV=@(input) P_ADMM(input,A,proj_z)
```

Workflow


model update direction
User provided code for FWI
gradient
function value
Optimization algorithm which handles projections
(projected & proximal algorithms)
vector
projected
vector
Projector onto set 1
Projector onto set 2
Projector onto set p
Algorithm to compute projection
onto intersection

Code(1)

User provides a code which computes function values and gradients:

```
%1) set up code to compute function value and gradient
data_misfit=@(input) compute_misfit_gradient(input,geometry,sources,frequencies);
```

This is a function handle which acts as:

```
[f,g]= data_misfit(m);
```

Code(2)

Use a script provided by the toolbox to get projectors onto each constraint set separately.
This script requires information about the model grid and possibly frequencyand initial model.

```
%2)obtain projectors onto each set separately
[Proj_bound,Proj_TV,Proj_rank] = setup_constraints(constraint,geometry,m0,frequencies);
```

Each projector is a function handle, input is projected onto the set:

```
output=Proj_TV(input)
```

Code(3)

Obtain the projector onto the intersection.

Requires the function handles to the separate projectors as input.

```
%3)set up Dykstra's algorithm
Proj_intersect = @(input) Dykstra(input,Proj_bound,Proj_TV,Proj_rank);
```

Output is the projection onto the intersection.

Code(4)

Call optimization algorithm using the data-misfit & gradient function handle and the intersection projector.

```
%4) optimize
m_est = SPG(data_misfit,m0,Proj_intersect);
```

SLIM

Code overview

```
%1) set up code to compute function value and gradient
data_misfit=@(input) compute_misfit_gradient(input,geometry,sources,frequencies);

%2)obtain projectors onto each set separately
[Proj_bound,Proj_TV,Proj_rank] = setup_constraints(constraint,geometry,m0,frequencies);

%3)set up Dykstra's algorithm
Proj_intersect = @(input) Dykstra(input,Proj_bound,Proj_TV,Proj_rank);

%4) optimize
m_est = SPG(data_misfit,m0,Proj_intersect);
```

Numerical examples

(projected) gradient descent is much too slow in practice.

Instead, we use (stochastic) versions of gradient descent with

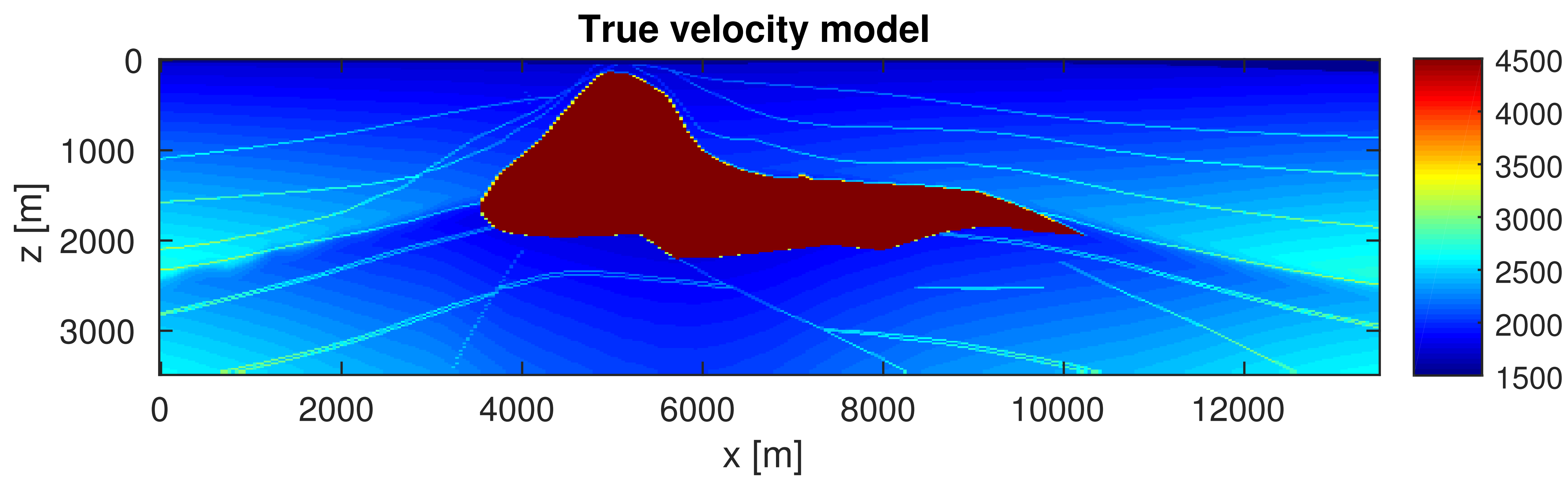
- non-monotone linesearch
- spectral scaling
- momentum/inertia.

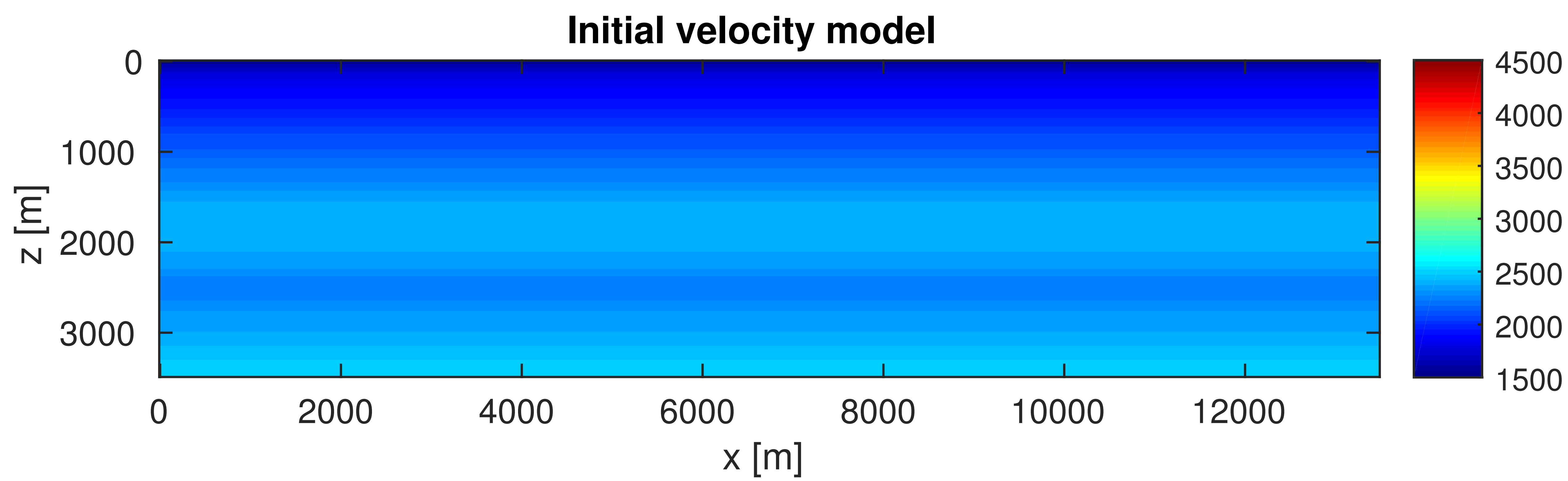
These methods are also less prone go get stuck in shallow local minimizers (empirically).

Numerical examples use a variant of (stochastic) non-monotone spectral projected gradient.

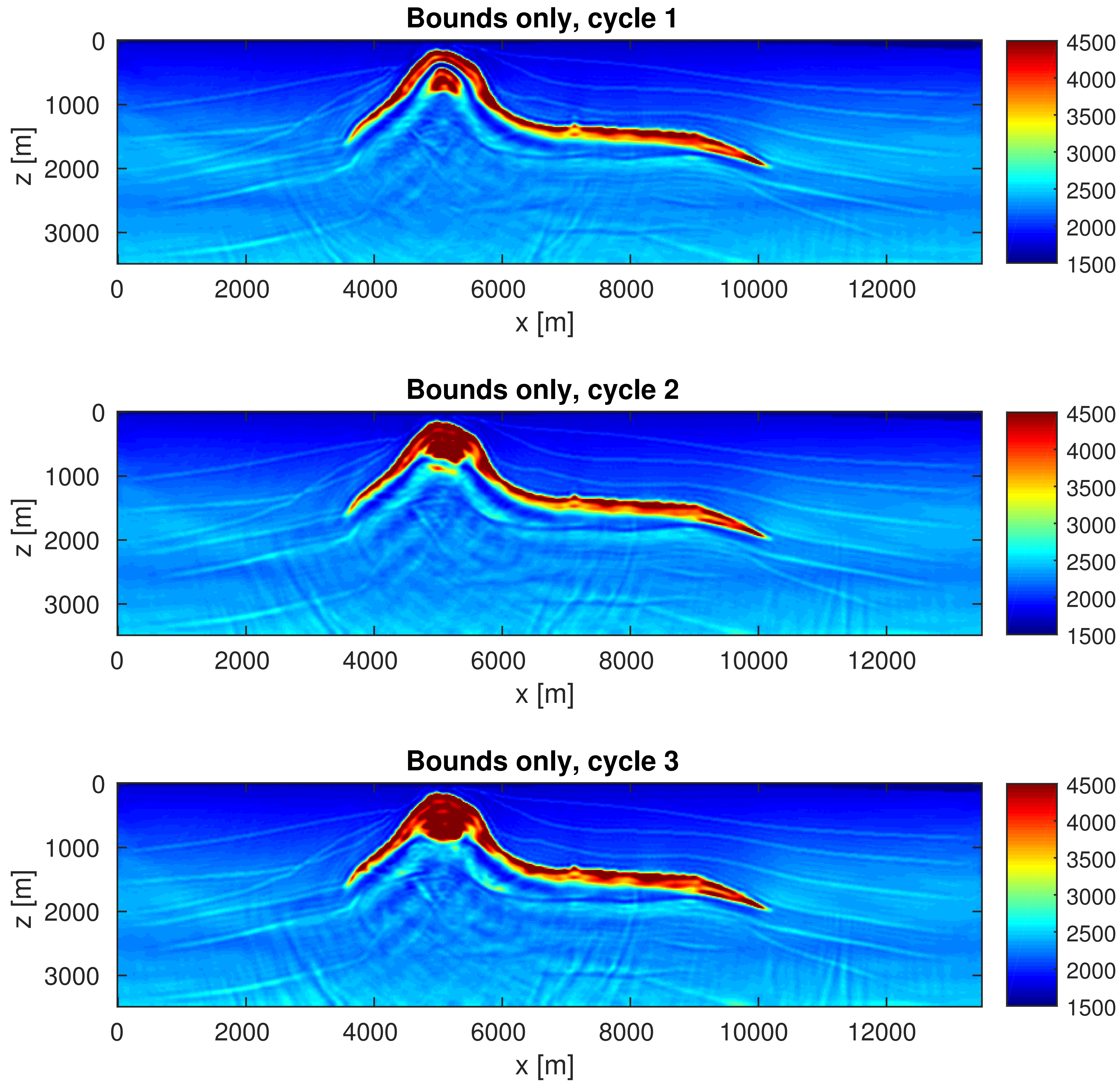
Frequency domain FWI example 3 – Salt structure

Frequency batches: $\{3, 3.33, 3.67, 4\}, \{4, 4.33, 4.67, 5\}, \{\ldots\}, \{12, 12.33, 12.67, 13\}$





Frequency domain FWI example 3 – Salt structure



Bound constraints only, restart 3 times

Frequency domain FWI example 3 – Salt structure

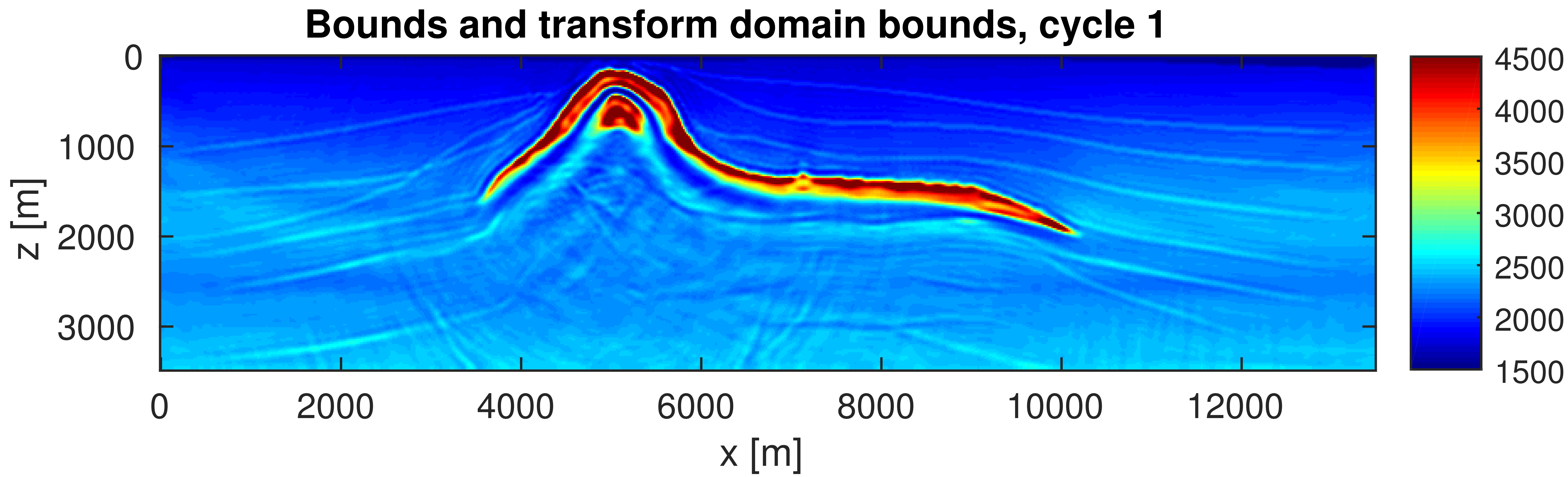
Questions:

- How can constraints help?
- Which constraints?
- How to select the associated parameters?

Frequency domain FWI example 3 – Salt structure

A possible strategy:

Only use bound constraints to see what works and what does not.



Observations:

- Top of salt looks good.
- Velocity drops down to minimum just below the top.

Frequency domain FWI example 3 – Salt structure

Need to prevent the velocity to drop quickly in the depth direction.

One option: pointwise slope constraints (as before):

$$\mathcal{C} \equiv \{\mathbf{m} \mid \mathbf{b}^l \leq A\mathbf{m} \leq \mathbf{b}^u\}$$

In this example $A\mathbf{m}$ means $\mathbf{b}_i^l \leq \dfrac{\mathbf{m}_{i+1,j} - \mathbf{m}_{i,j}}{h_z} \leq \infty$

In words:

velocity can increase with depth, unbounded.

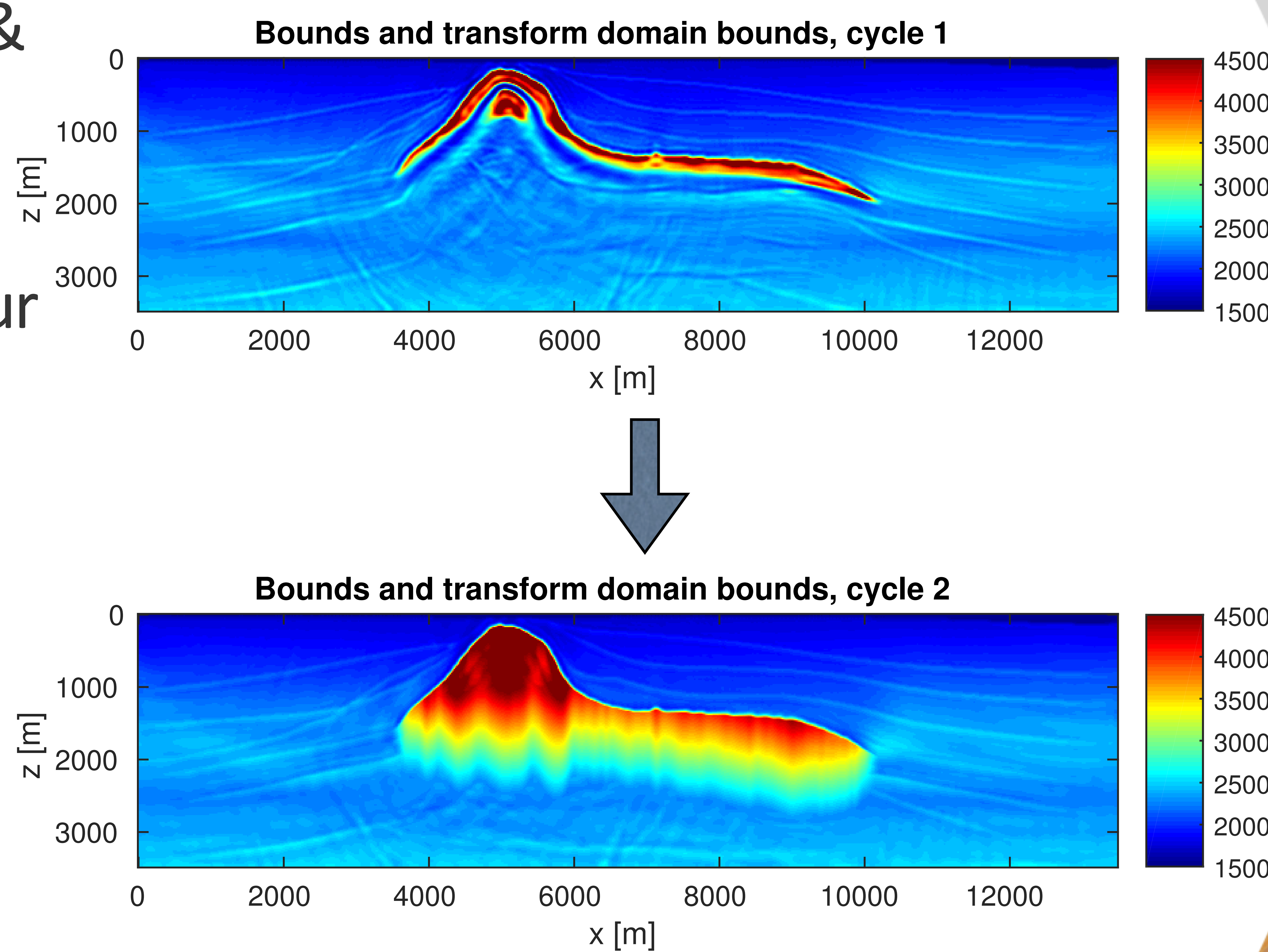
velocity can only slowly decrease with depth. $\mathbf{b}_i^l$ = small negative number

Frequency domain FWI example 3 – Salt structure

Use previous result as initial guess & rerun inversion.

Salt is extended downwards, but our current constraints do now allow a sharp salt bottom.

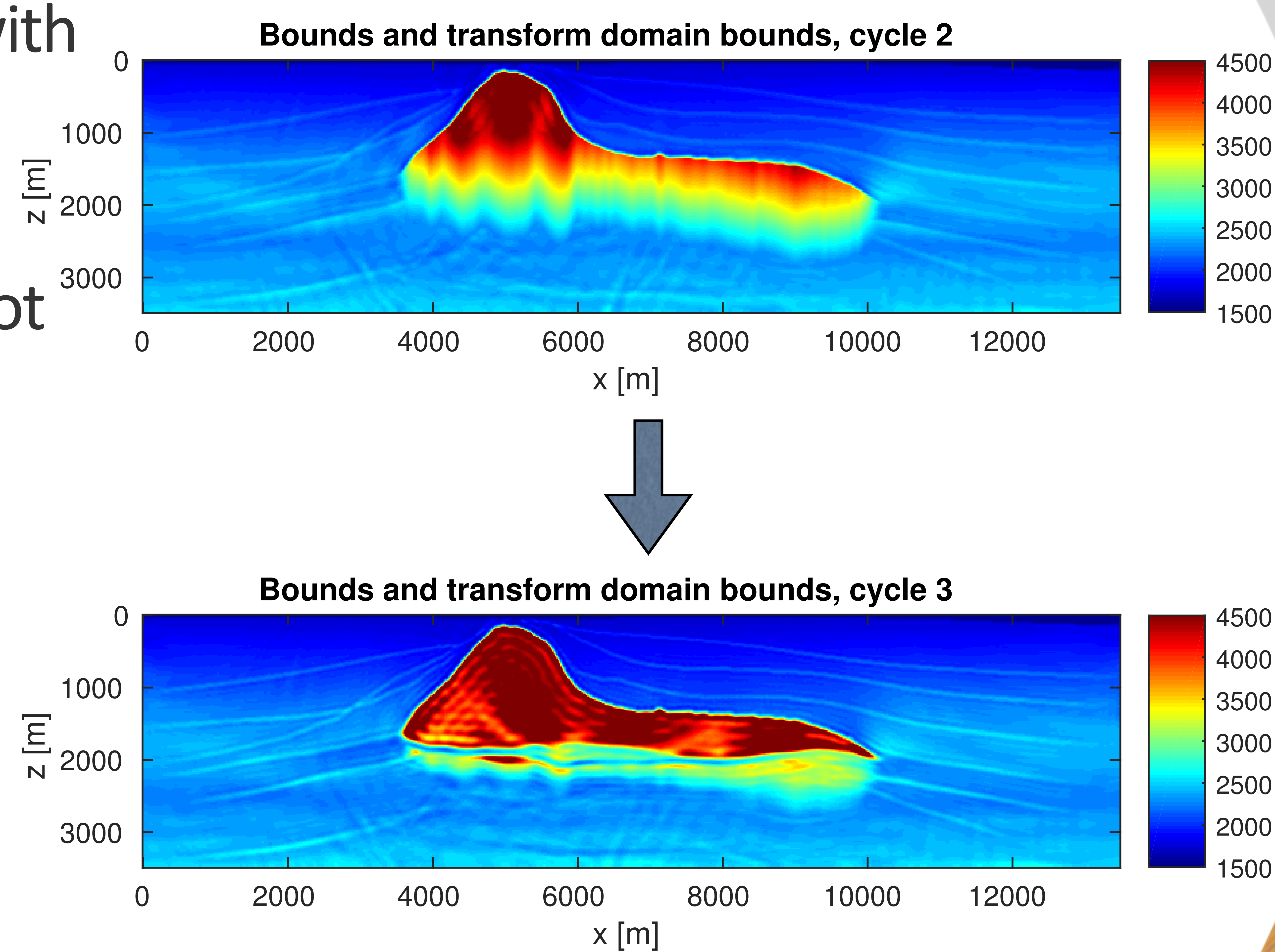
Solution...



Frequency domain FWI example 3 – Salt structure

Turn of slope constraints & rerun with just bound constraints.

Salt bottom is much sharper, but not perfect.



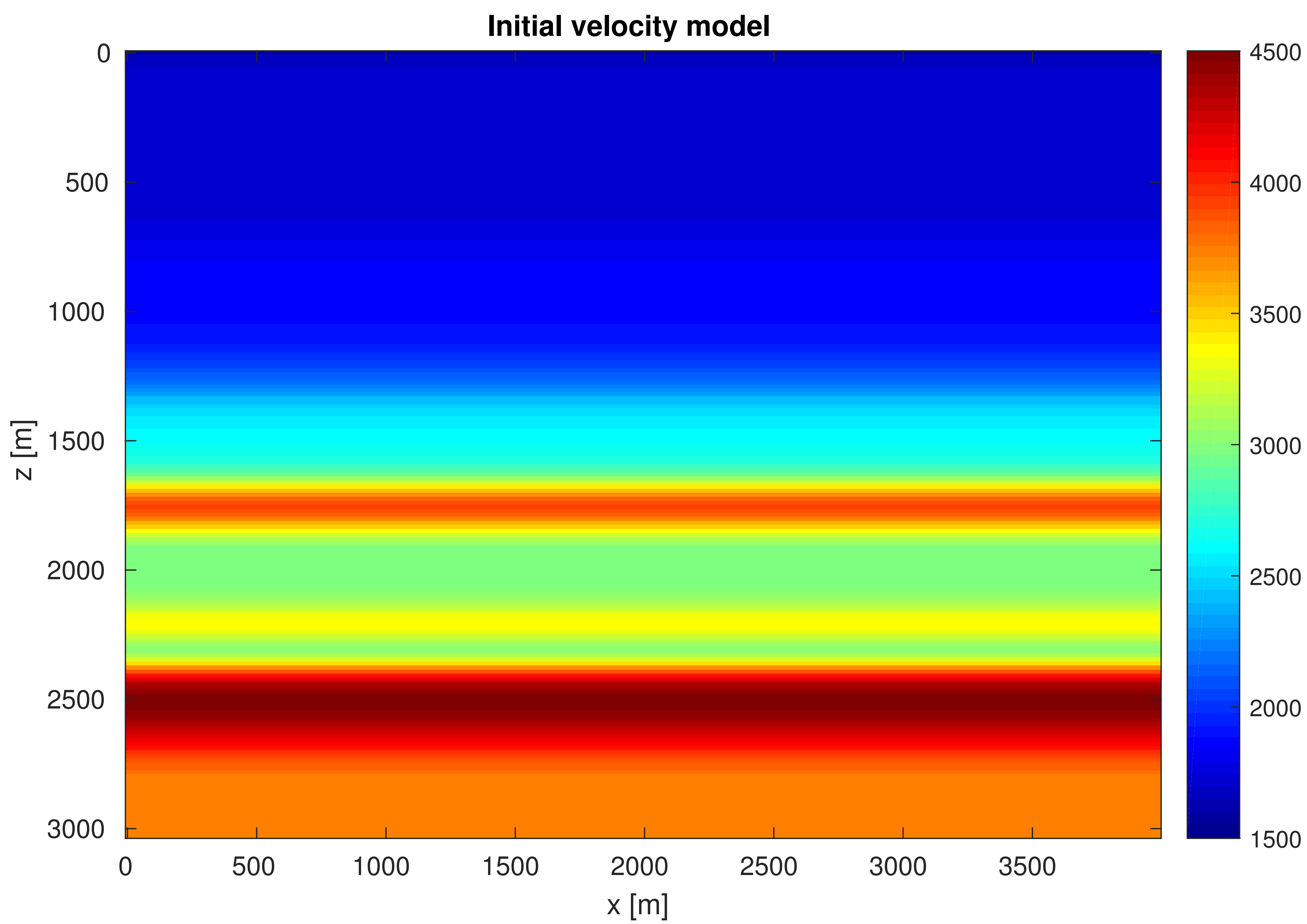
Frequency domain FWI example 4 – sediments

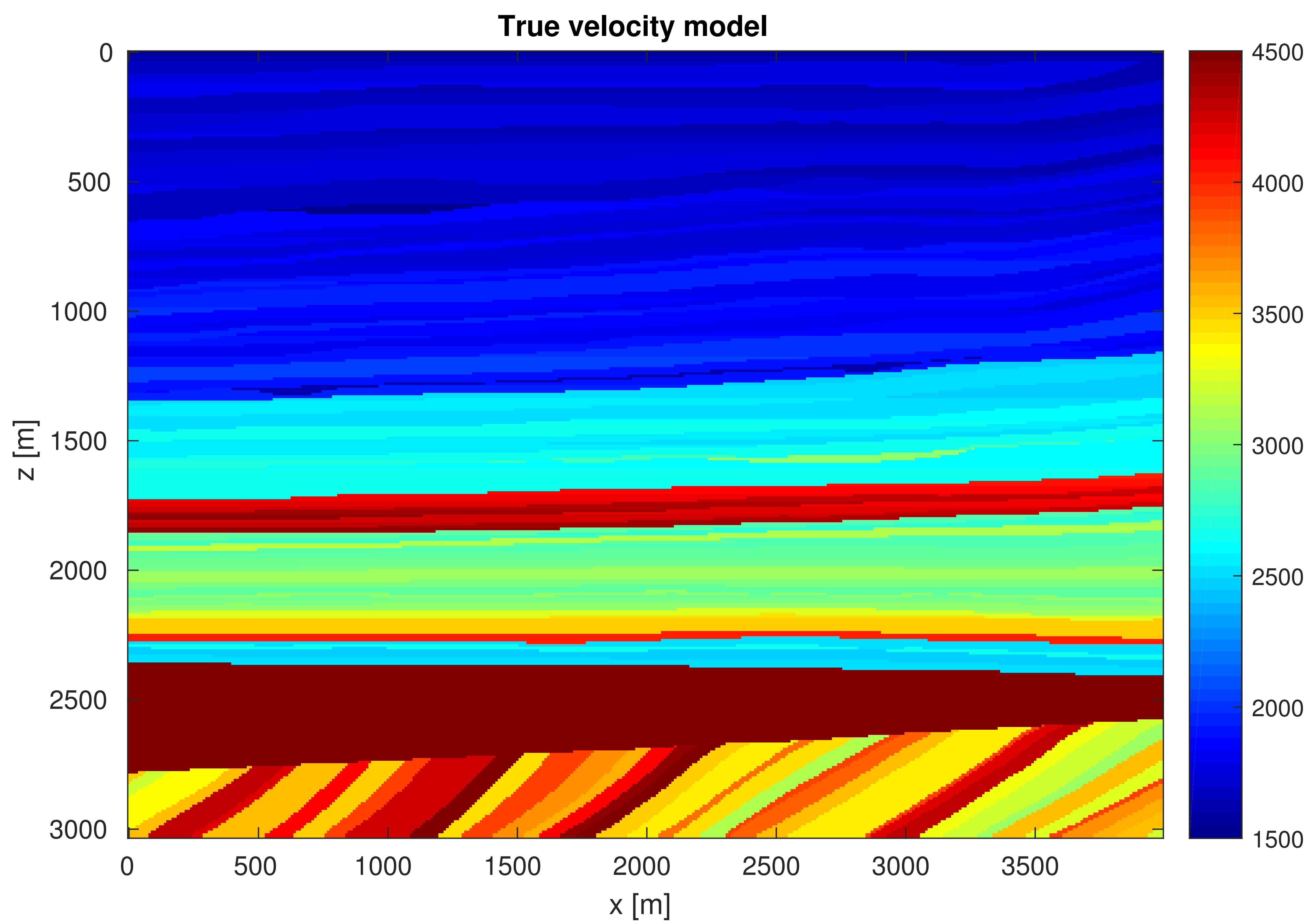
Another strategy if we do not have much prior information:

- Combine many 'weak' constraints.
- Each constraint eliminates a class of physically unrealistic models.
- The intersection is then a more 'powerful' constraint set.
- Philosophy: describe what the model should not look like.

Frequency domain FWI example 4 – sediments

left side of the Marmousi model
10 simultaneous sources
zero mean Gaussian noise

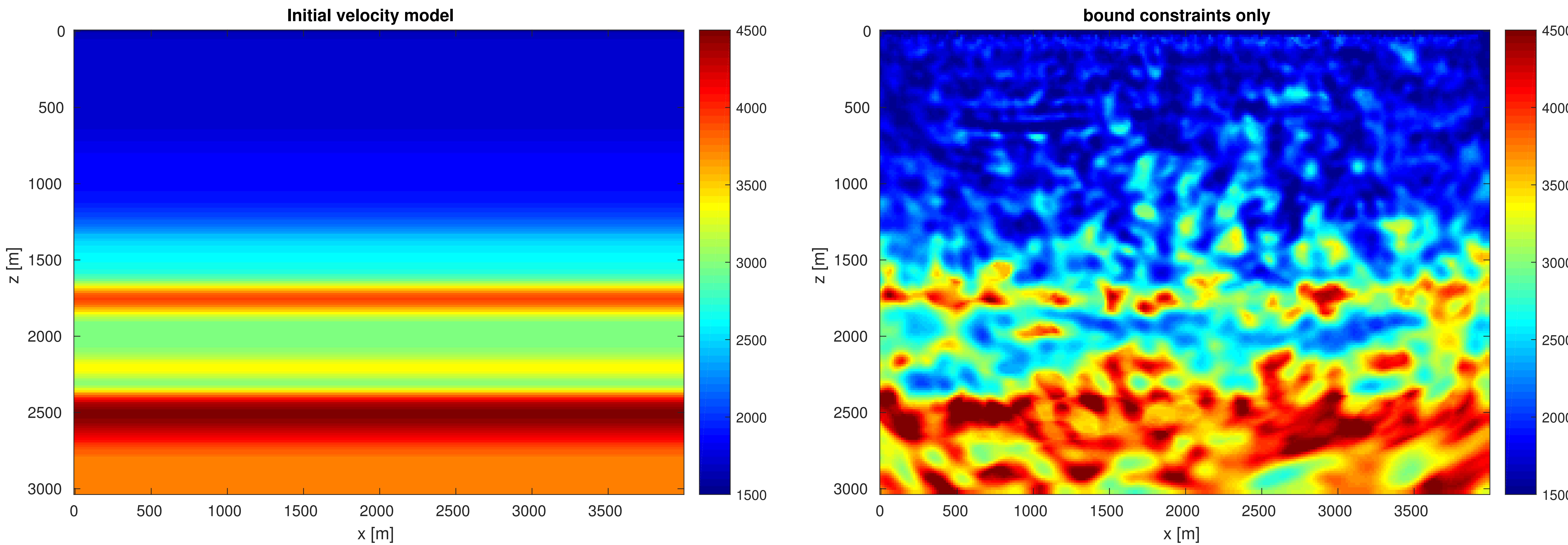




Frequency domain FWI example 4 – sediments

left side of the Marmousi model
10 simultaneous sources
zero mean Gaussian noise

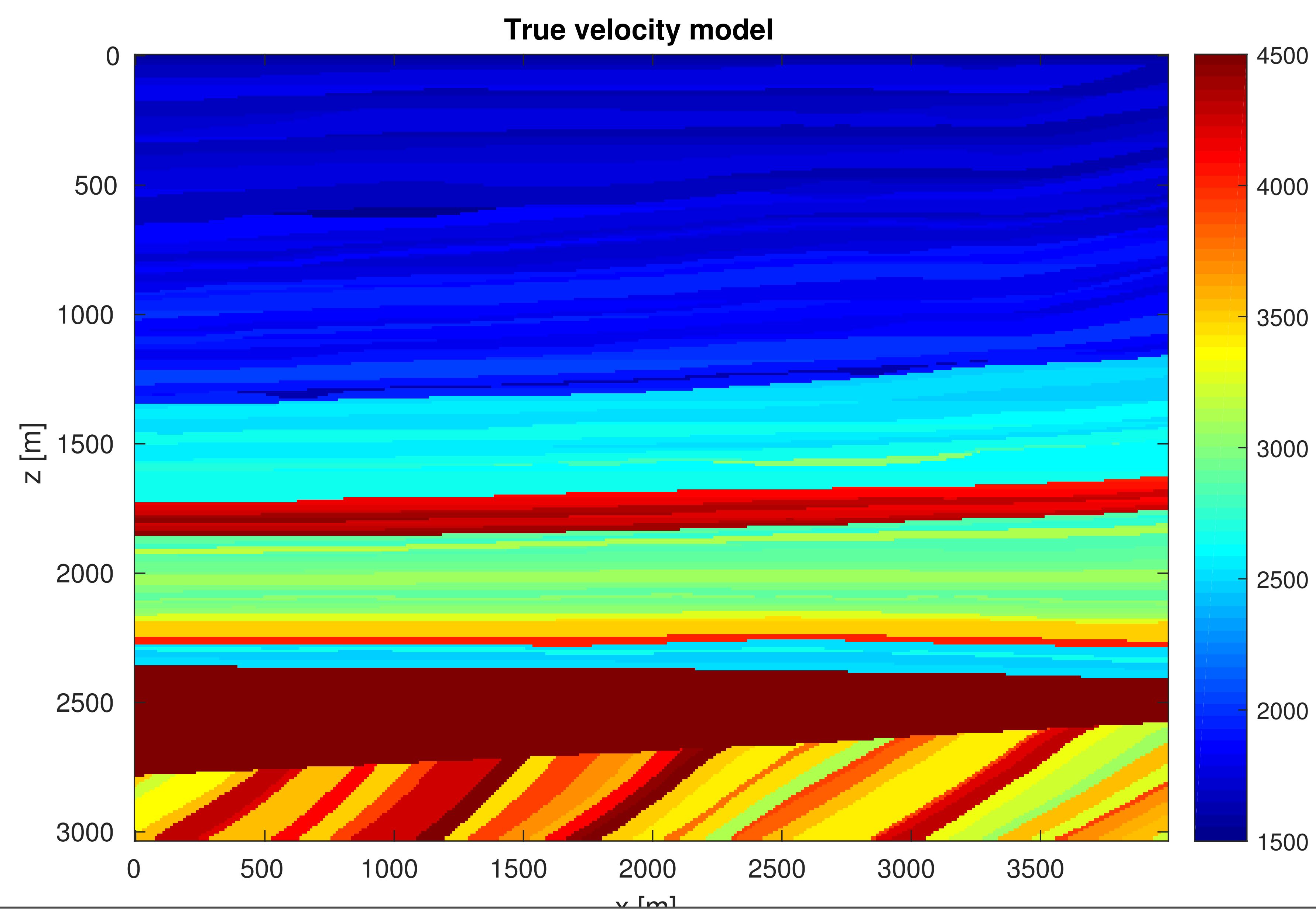
Bound constraints only



Frequency domain FWI example 4 – sediments

Now use all of the prior information below:

1. Bound constraints
2. Reasonable initial model -> tighten bounds to start model +/- 1000 m/s
3. Coarse structure is blocky -> limit the total-variation a little bit
4. Approximately layered, except the bottom -> rank constraint
5. No large jumps in the horizontal direction -> slope constraint on horizontal gradient.
6. Small number of horizontal velocity jumps->limit cardinality of horizontal gradient.

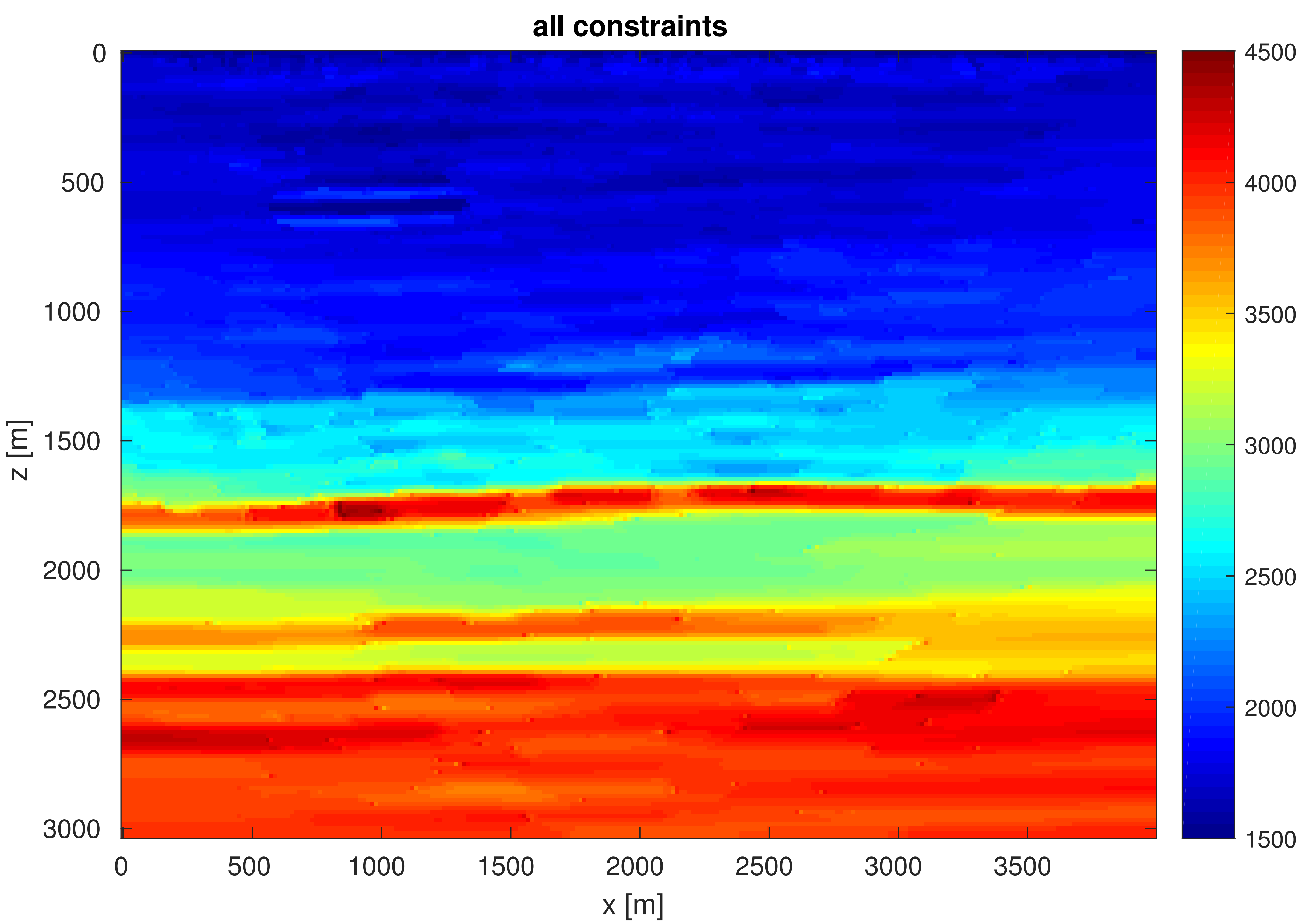


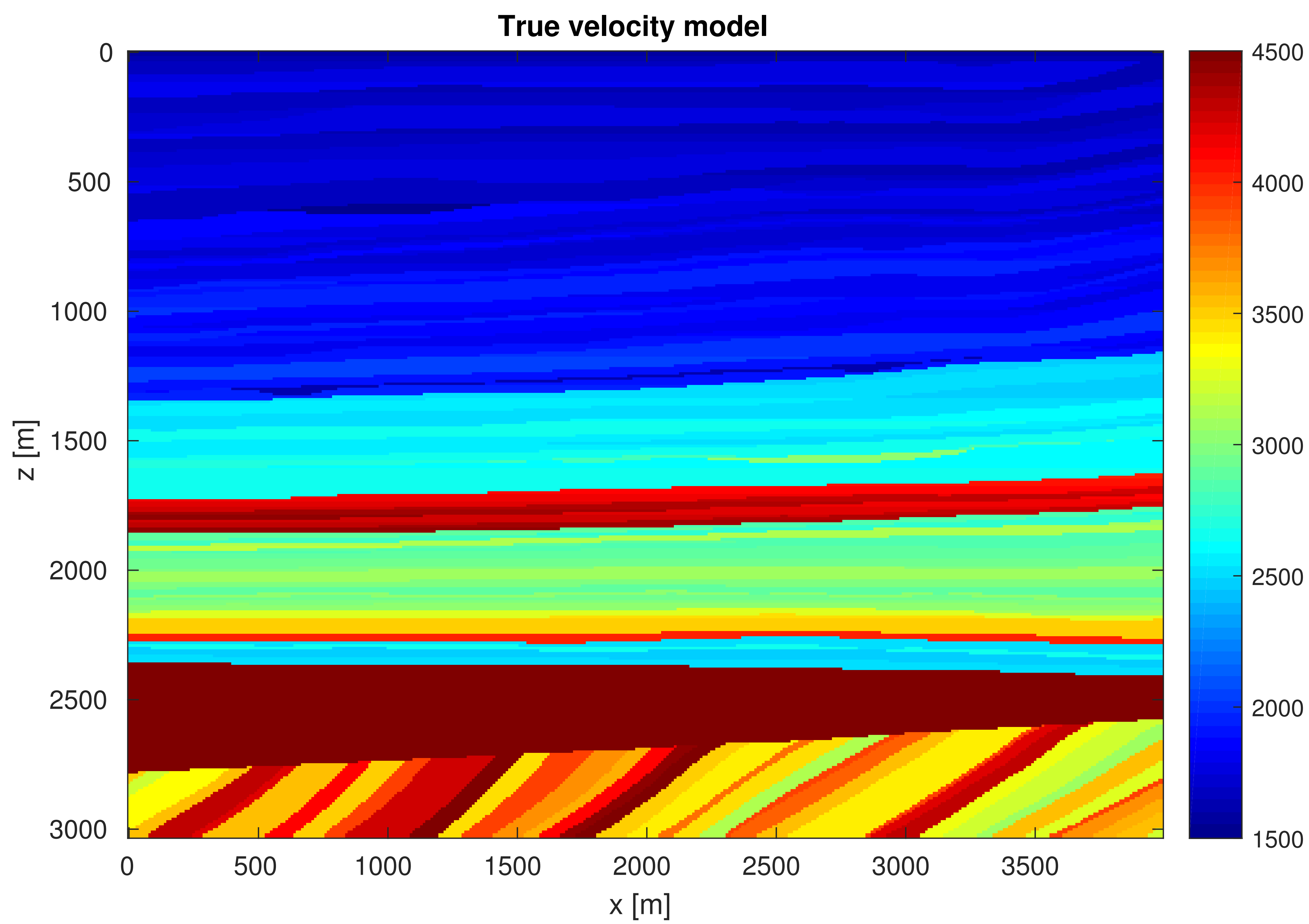
Frequency domain FWI example 4 – sediments

Much better model estimate.

Bottom part not estimated well, because it was not described by the constraints.

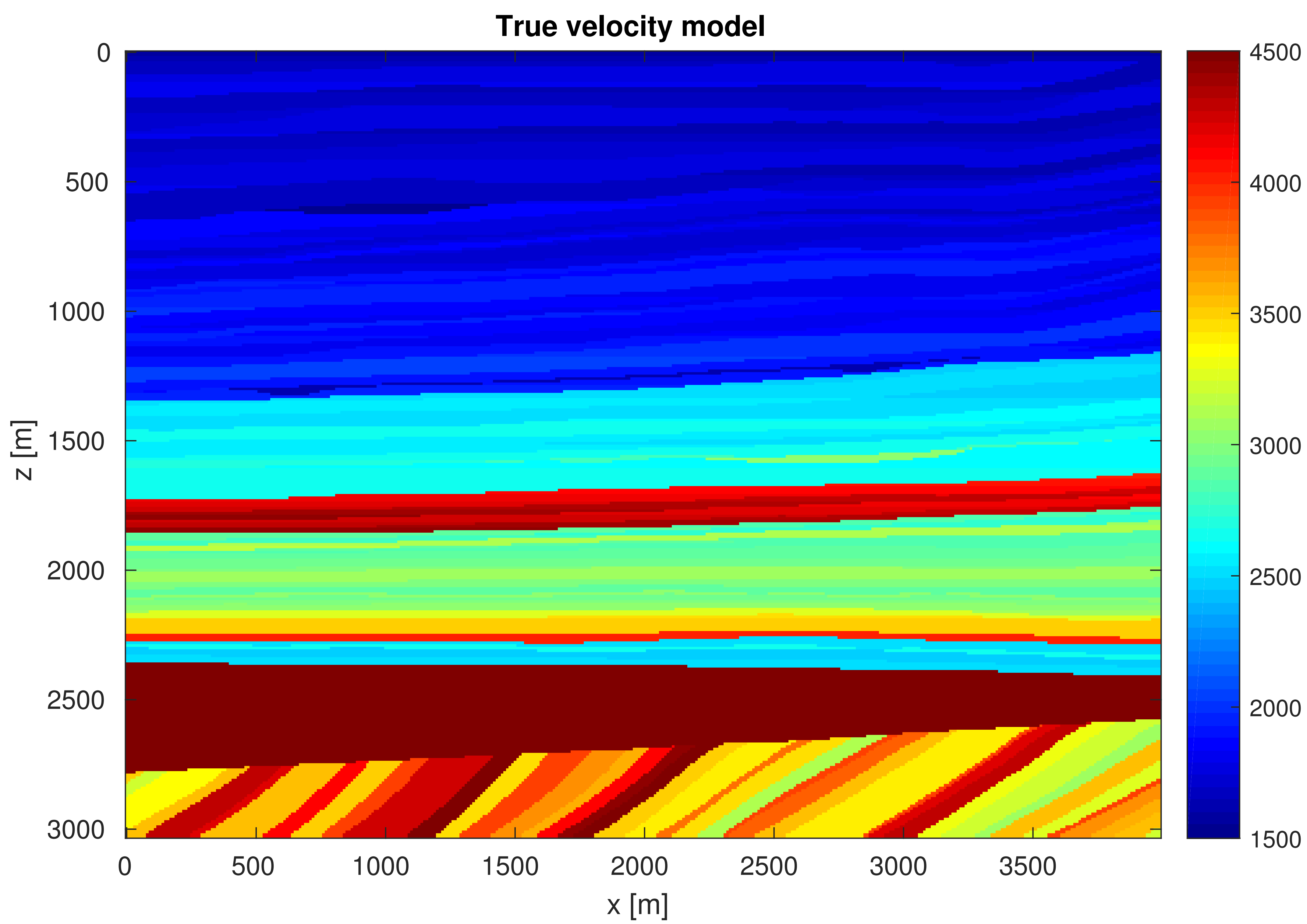
All constraints simultaneously





Frequency domain FWI example 4 – sediments

True model



Convex vs non-convex sets

convex

pro:

- Dykstra & ADMM will converge
- Any other algorithms can be swapped in and it will work as well.

con:

- Constraint definition sometimes not intuitive or difficult to estimate.

Convex vs non-convex sets

non-convex

pro:

- Constraint definition is often more intuitive.

con:

- Dykstra & ADMM may not find projections, but approximations.
- Any other algorithm needs to be carefully tested for robustness in case of non-convex sets.

Related geophysical work (1)

[A. Baumstein, 2013] . This work attempts to find the projection onto an intersection using POCS, for different constraints. Includes preconditioner in the projected gradient algorithm. May not converge.

[E. Esser et. al., 2014; 2015] (UBC Tech report; EAGE 2015). Similar philosophy/ideas & problem formulation, different constraints and algorithms.

[B. Peters, B. R. Smithyman & F.J. Herrmann, 2015] (UBC Tech report) projected quasi-Newton based version of this presentation.

[B. R. Smithyman, B. Peters & F.J. Herrmann, 2015] (EAGE,2015). About real land dataset, uses projected quasi-Newton.

[S. Becker et. al., 2015]. (EAGE,2015) Also uses projected/proximal quasi-Newton, for projections onto a single set. Curvelet domain sparsity/TV.

[B. Peters, Z. Fang, B. R. Smithyman & F.J. Herrmann, 2015] (submitted to SEG 2015 conference). About the Chevron blind-test dataset (2014). Projected Newton-type using ADMM.

Related geophysical work (2)

Conclusions & remarks

Non-convex sets may be easier to use.

Current algorithms perform sufficiently well with non-convex sets, if combined with other sets.

Working with non-convex sets is an active research topic, expect improved robustness in the near future.

Suitable for any nonlinear inverse problem.

All software is available at our Github (Matlab).

Almost finished: compiled Matlab, variables passed as filenames.

Acknowledgements

This research was carried out as part of the SINBAD project with the support of the member organizations of the SINBAD Consortium.