Algorithms and Julia software for constrained FWI

Bas Peters
SINBAD FALL 2017

SLIM
University of British Columbia

Motivation

Develop constraints & optimization methods to deal with:

- noisy data
- inaccurate starting models
- small number of data points

Constraints encode information about:

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

Goal

Create software toolbox that builds on top of existing codes:

- use any code for data-misfit value and gradient
- for inverse problems with expensive function & gradient
- arbitrary combinations of convex and non-convex sets
- all iterates satisfy all constraints
- convenient translation of prior information into constraints
- data-misfit and constraints are decoupled
- no penalty functions & parameters

SLIM

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

Transform-domain bounds / slope constraints

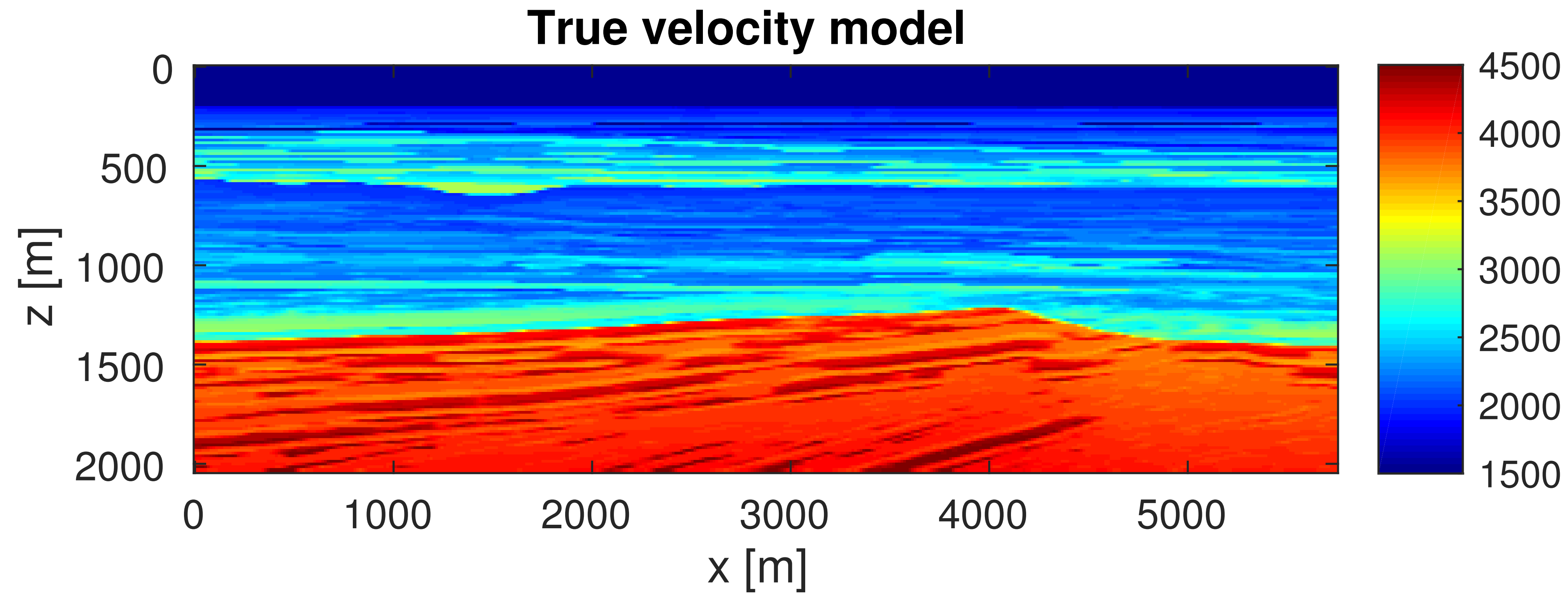
$$\mathcal{C} \equiv \{\mathbf{m} \mid \mathbf{l}_j \leq (A\mathbf{m})_j \leq \mathbf{u}_j\}$$

slope constraint if: $A = I_x \otimes D_z$ with $D_z = \frac{1}{h_z}\begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$

Interpretation:

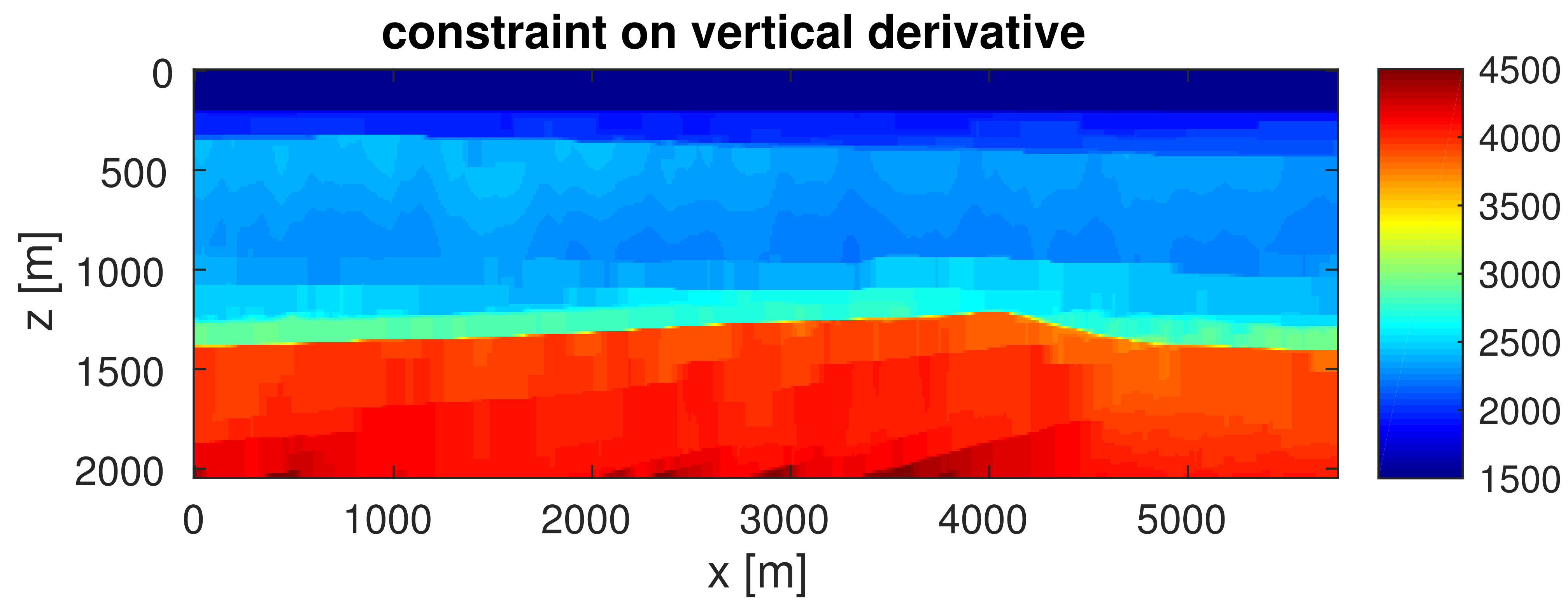
- limit the medium parameter variation per distance unit
- select different bounds for increasing values and decreasing values

Transform-domain bound constraints



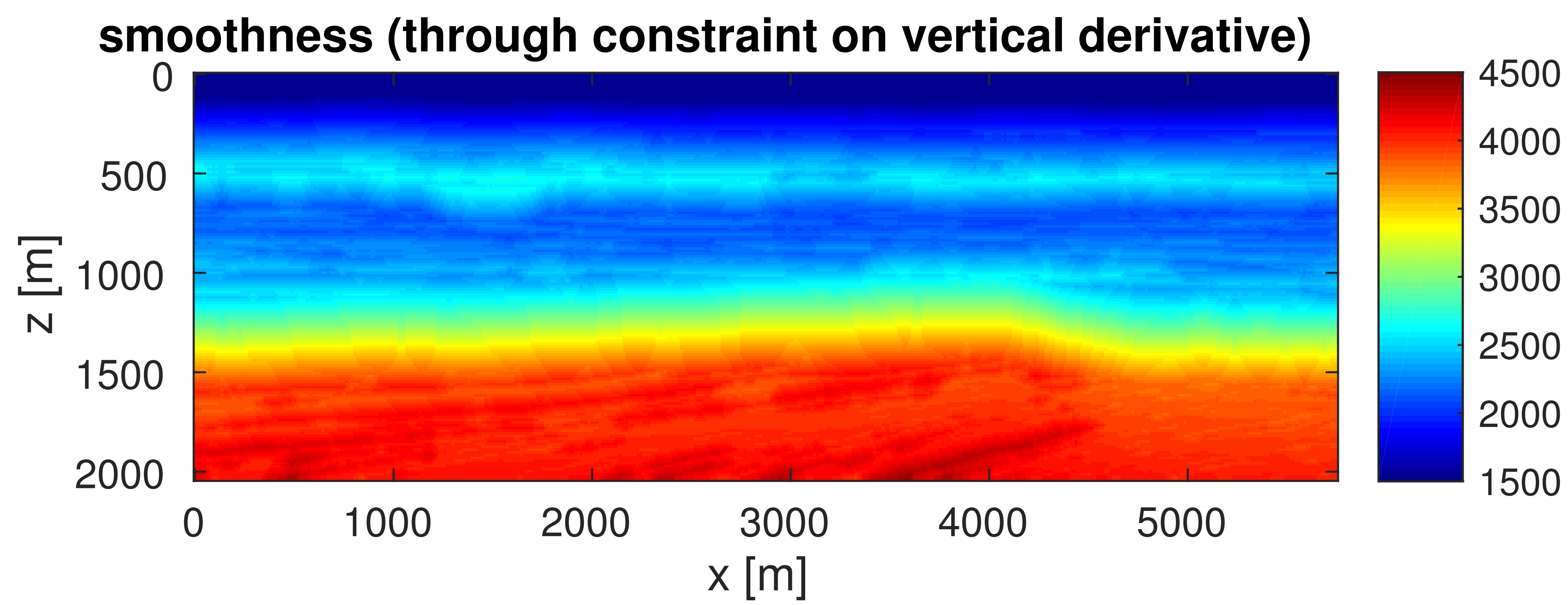
arbitrary medium parameter increase, limited medium parameter decrease with depth

- **induces monotonicity**



limited increase and limited decrease

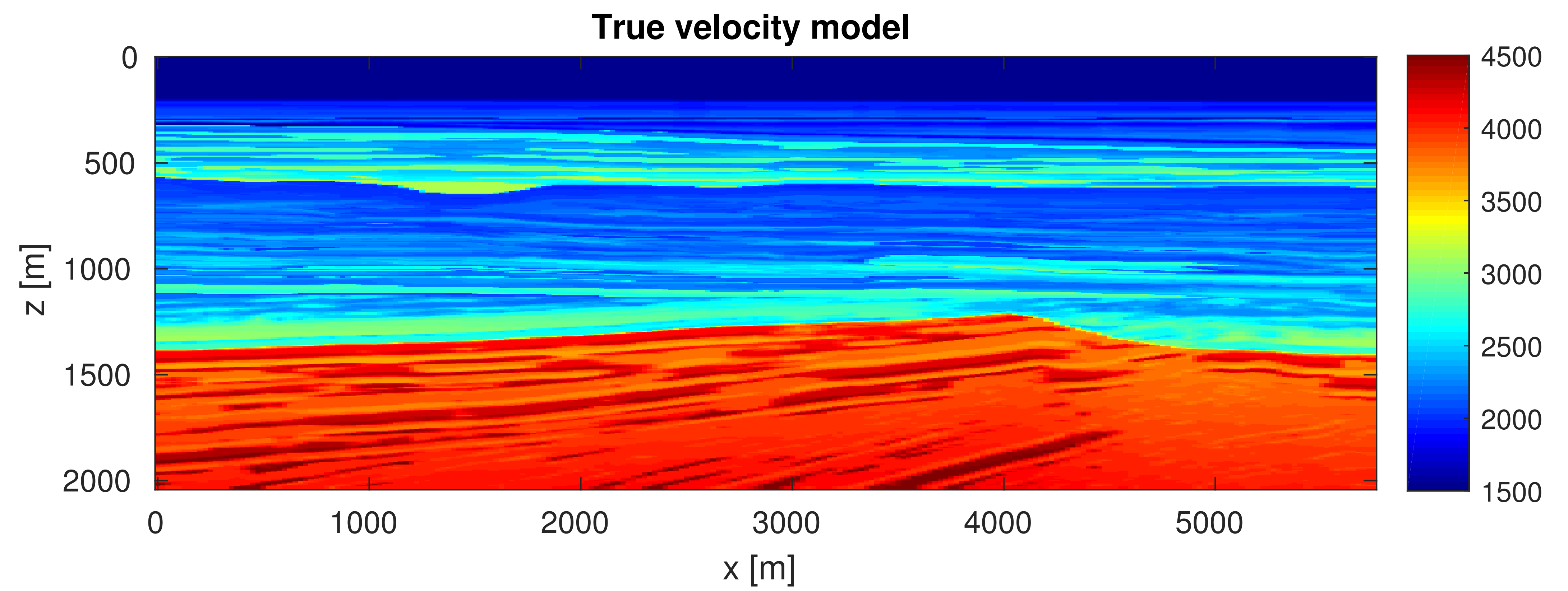
- **induces vertical smoothness**
- **still allows small velocity jumps**



Example - BG Compass

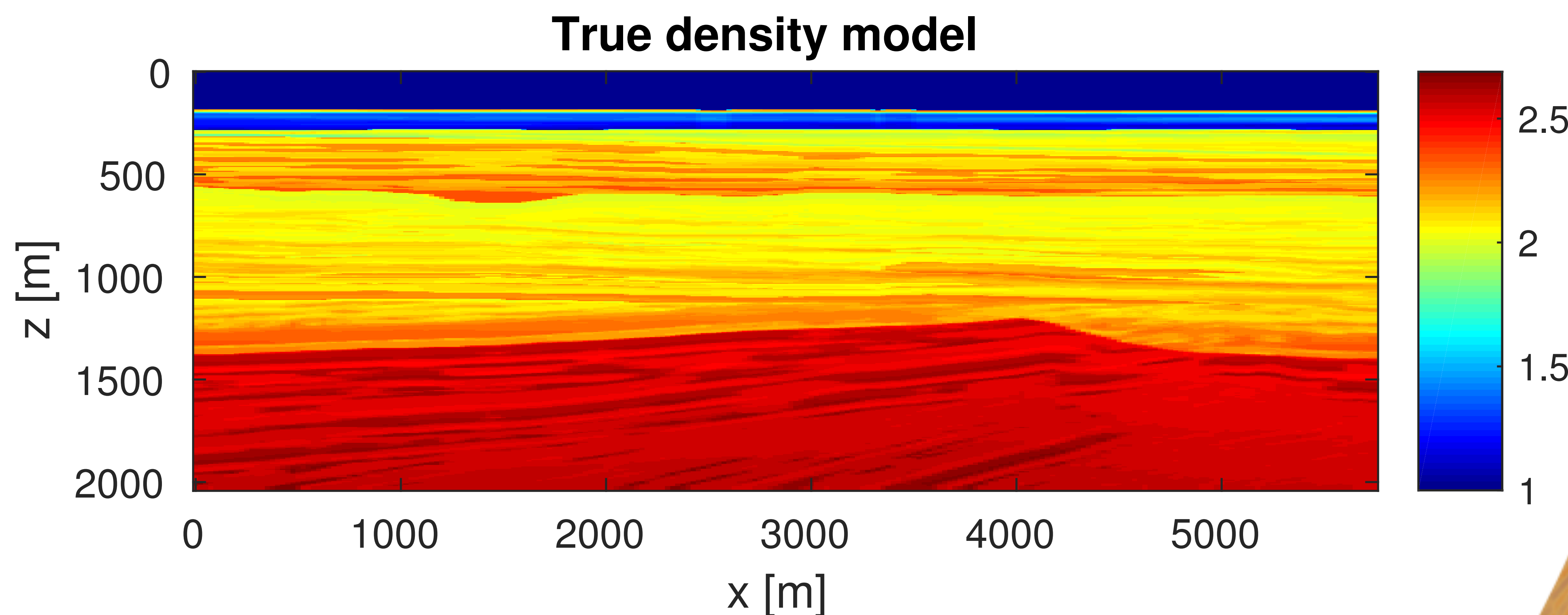
modeling 'observed' data:

- generate data on original 6m grid
- time-domain modeling (Julia interface for Devito **[Lange et. al., 2017]**)
- density and velocity



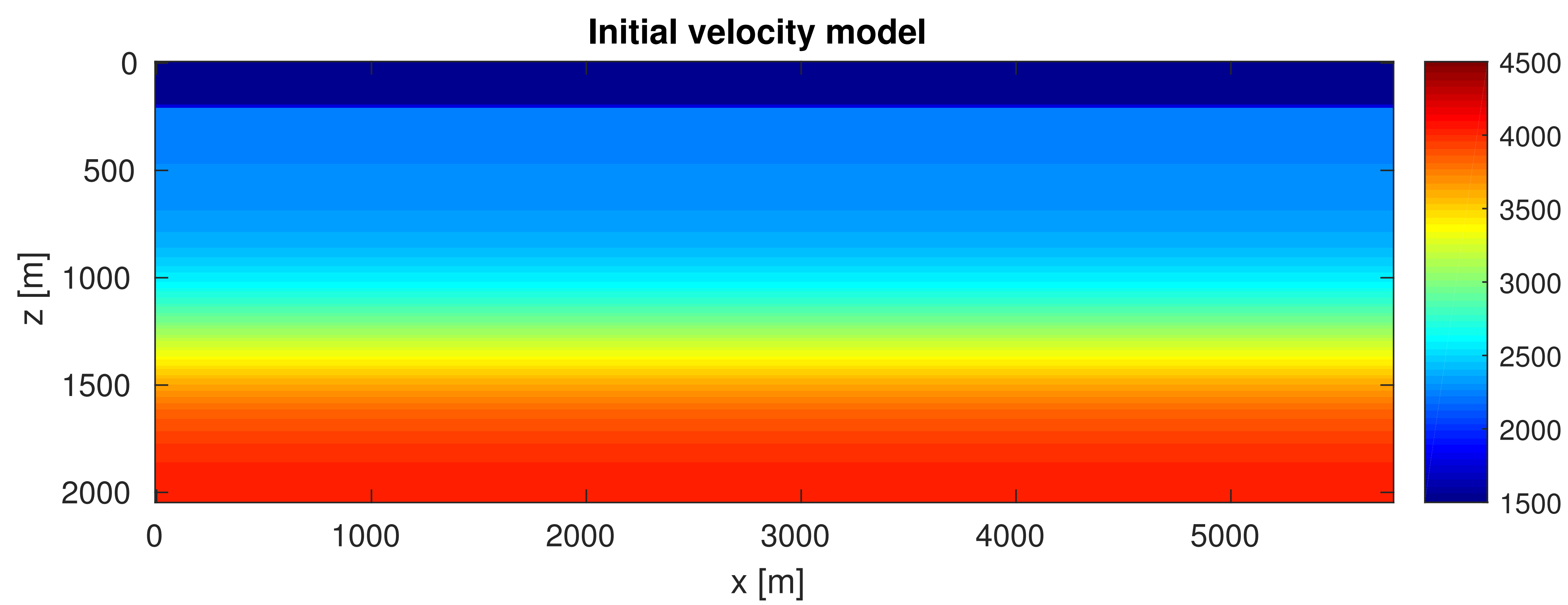
inversion

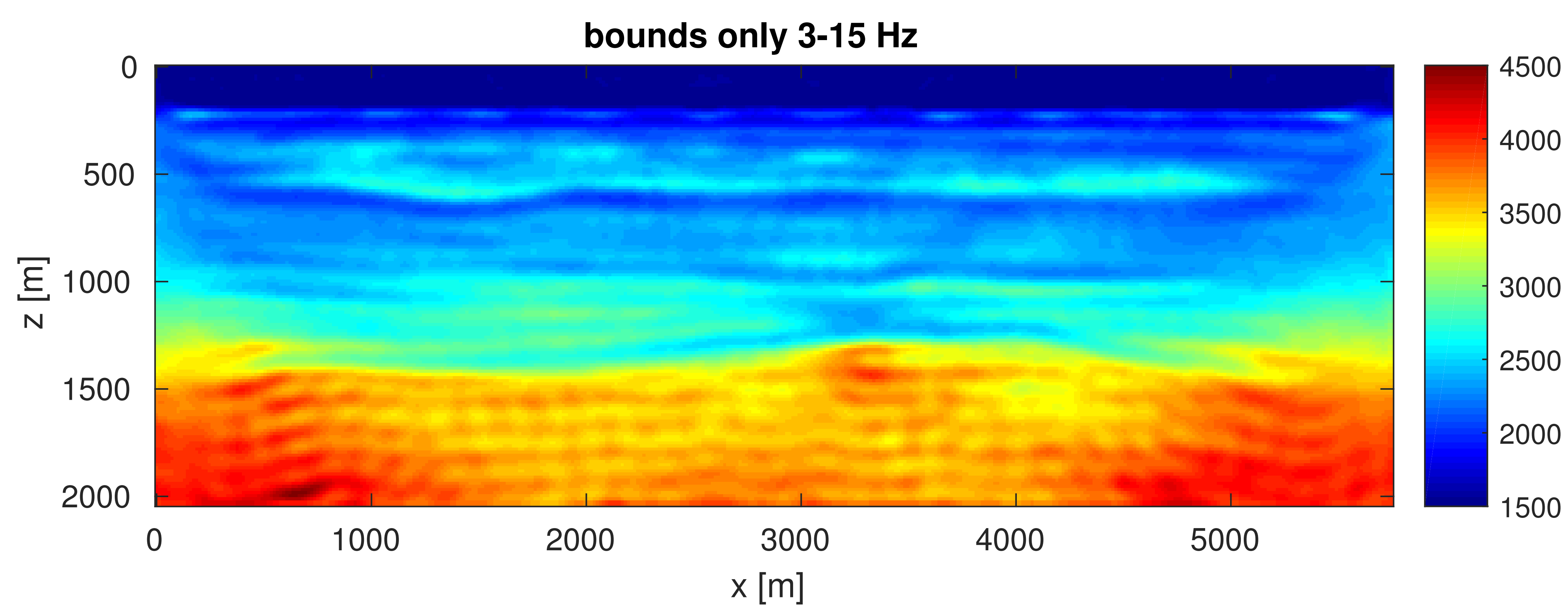
- for velocity only
- fixed density = 1 **[Da Silva & Herrmann, 2017]**
- frequency domain package *WAVEFORM*
- adapt grid for each frequency
- start at 60m grid ⟶ 15m grid



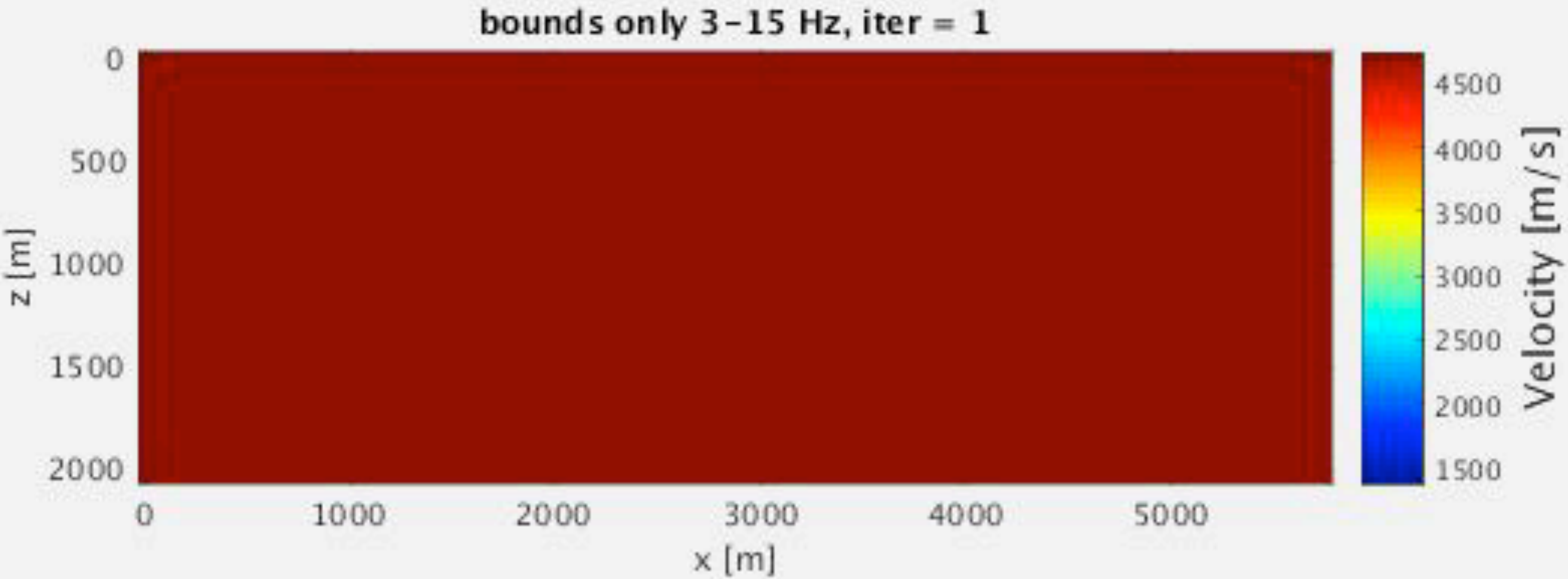
Example - BG Compass

- 3-15 Hz data, in 1Hz batches from low to high frequency
- bound constraints





Example - BG Compass



Example - BG Compass

So far, we (SLIM) used constraints to describe true model.

[E. Esser et. al., 2014; 2015; 2016] [Peters & Herrmann, 2017][B. R. Smithyman, B. Peters & F.J. Herrmann, 2015]

What if we do not know much about expected model?

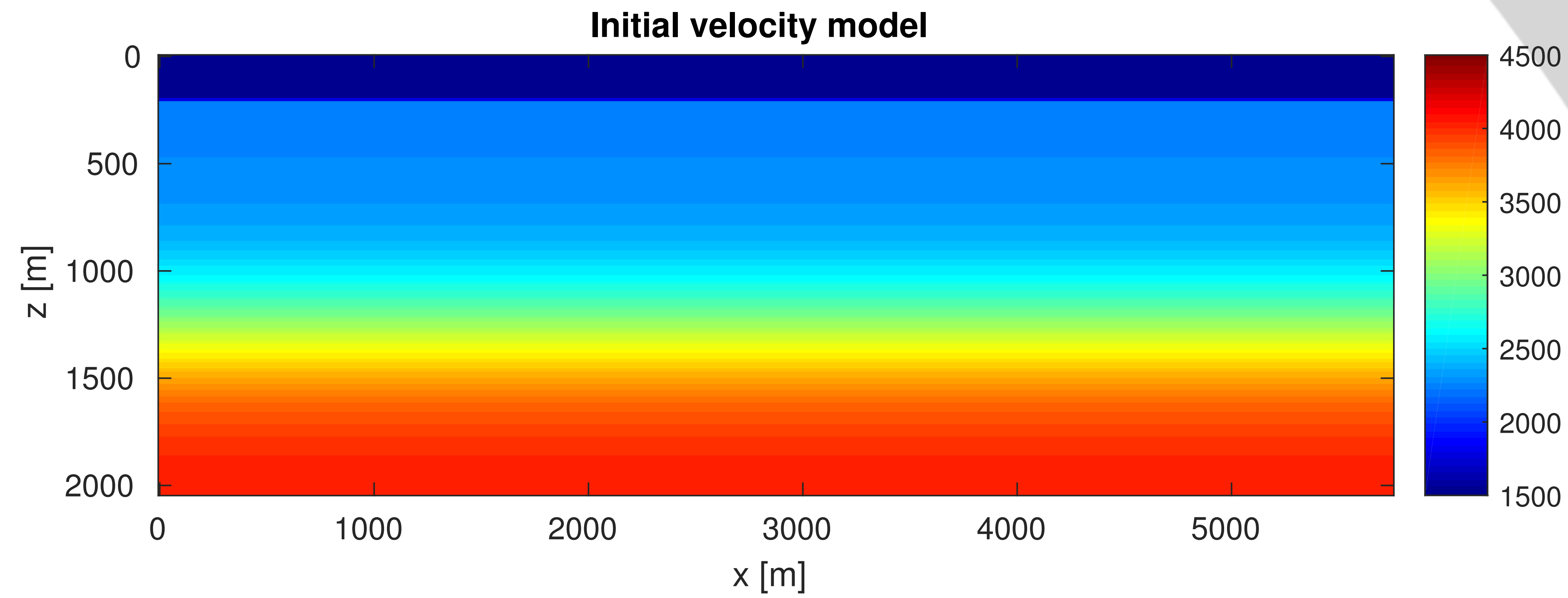
→ use constraints to obtain better starting model

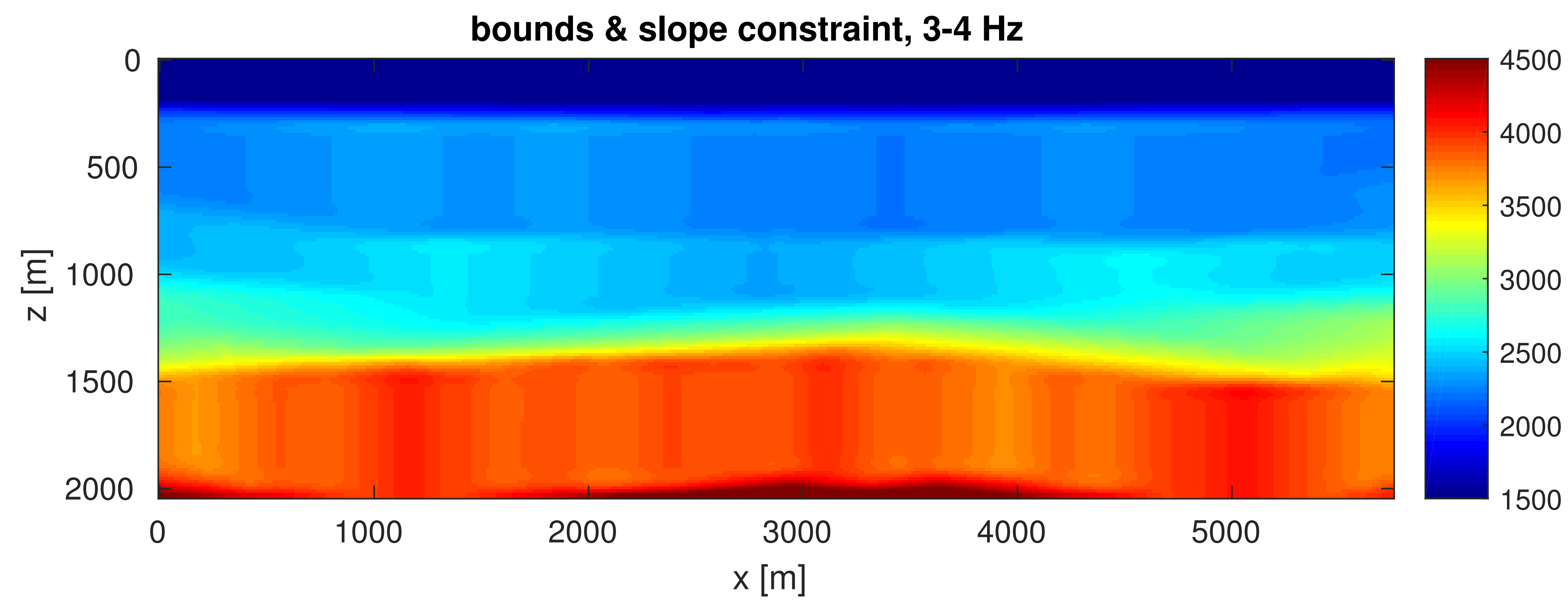
prior assumptions:

- sedimentary geology, mainly layered, no big faults
- starting model should be laterally smooth & velocity increases with depth

1st cycle: invert 3-4 Hz data with:

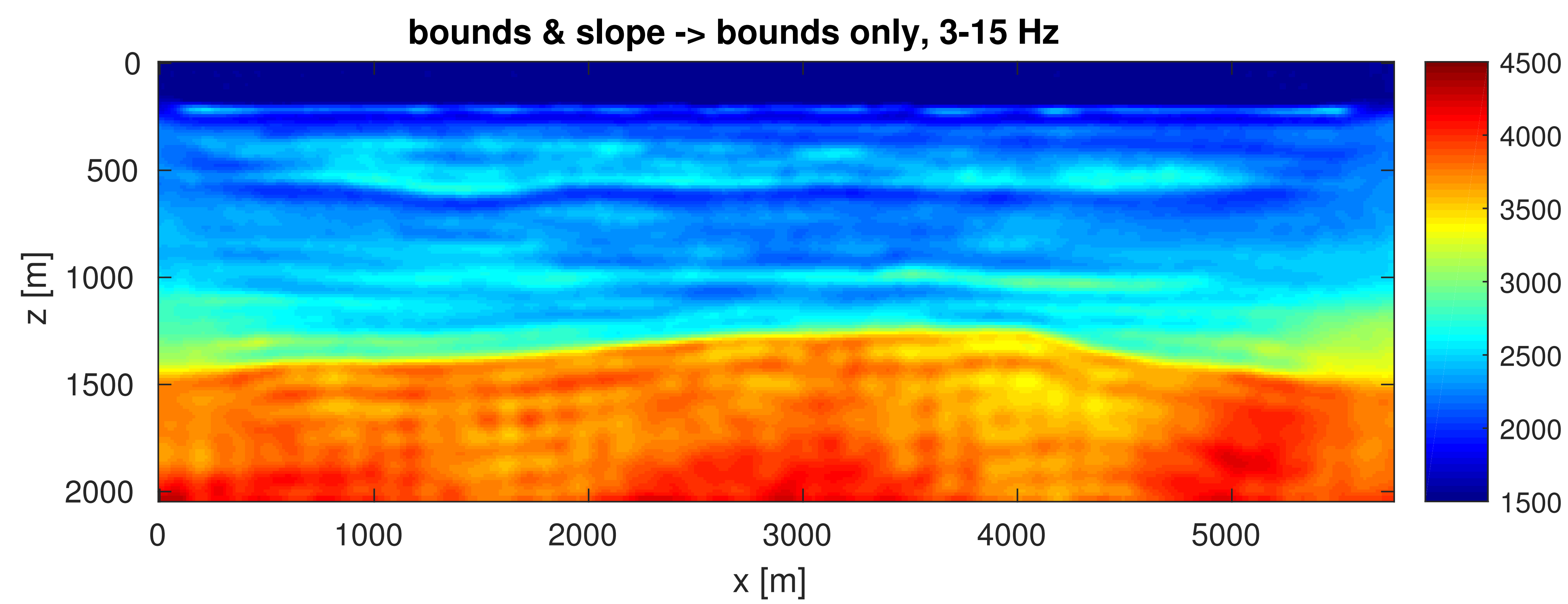
- bound constraints
- lateral smoothness (slope constraint):
  $$\{\mathbf{m} \mid -\varepsilon_1 \leq ((I_z \otimes D_x)\mathbf{m})_j \leq +\varepsilon_2\}$$
- approximate vertical monotonicity:
  $$\{\mathbf{m} \mid -\varepsilon \leq ((D_z \otimes I_x)\mathbf{m})_j \leq +\infty\}$$



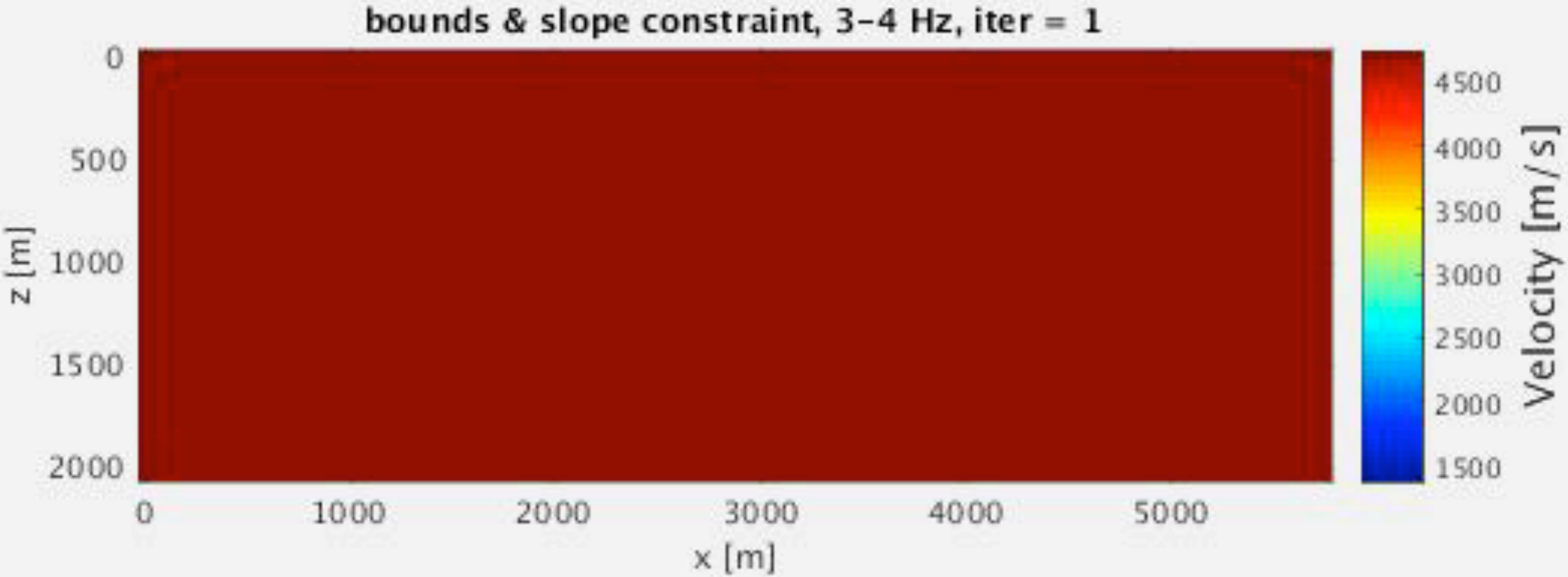


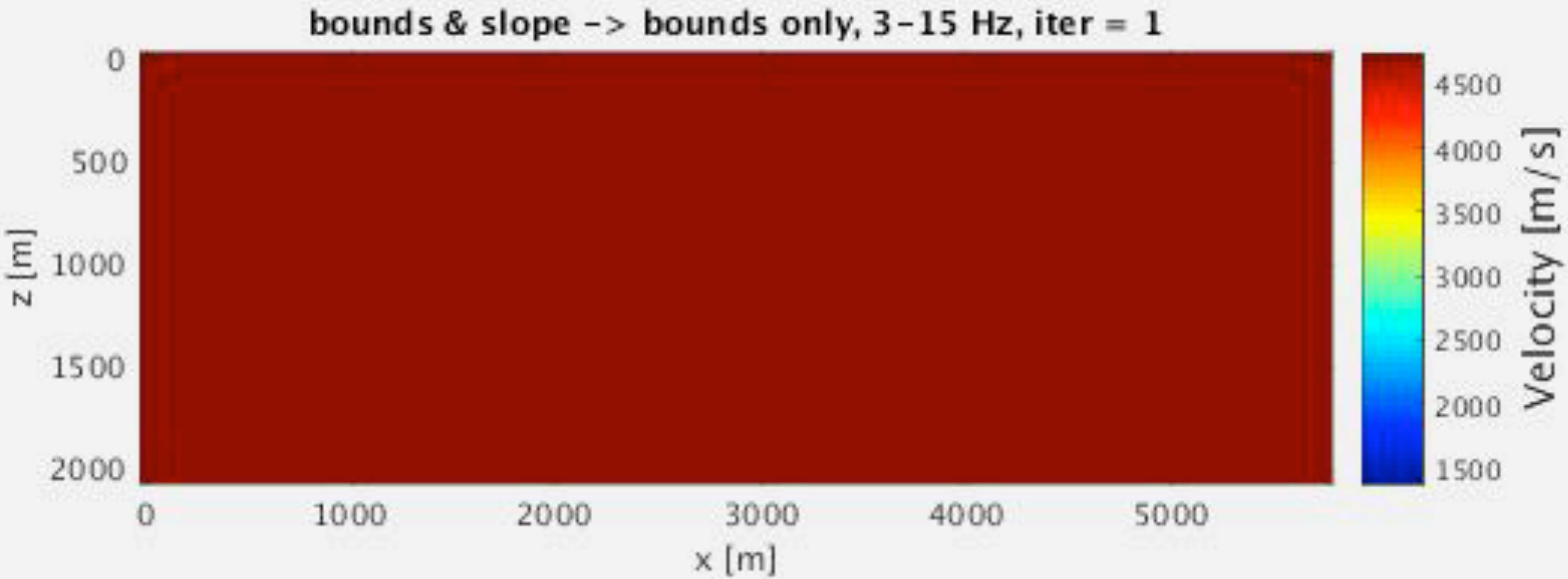
2nd cycle:

- use 1st cycle result as new starting model
- invert all data with bound constraints



With constraints, cycle 1




bounds & slope -> bounds only, 3-15 Hz, iter = 1
0
500
1000
1500
2000
z [m]
0
1000
2000
3000
4000
5000
x [m]
4500
4000
3500
3000
2500
2000
1500
Velocity [m/s]

Problem formulation

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

differentiable data-misfit   intersection of constraint sets

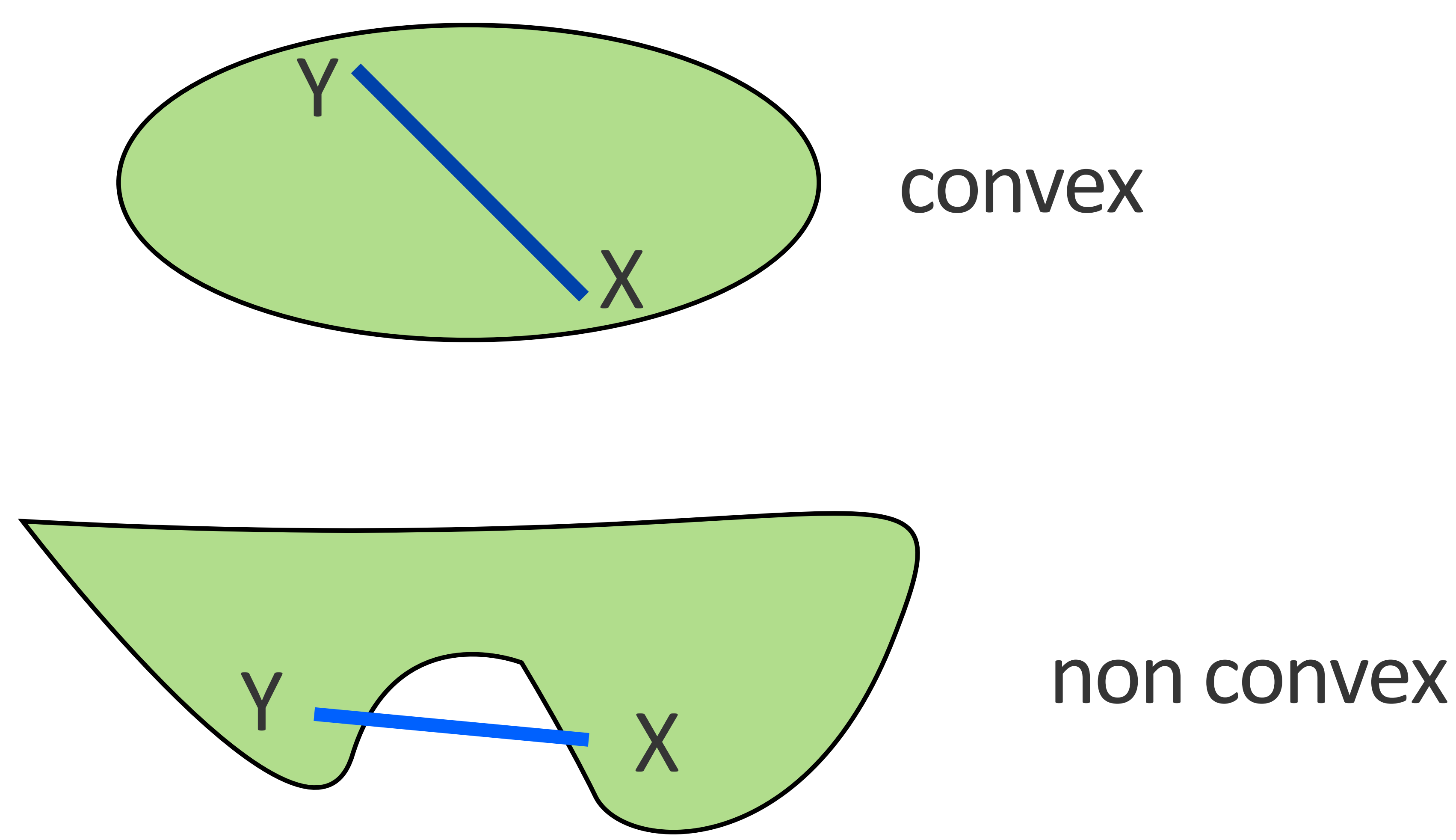
Geophysical applications:

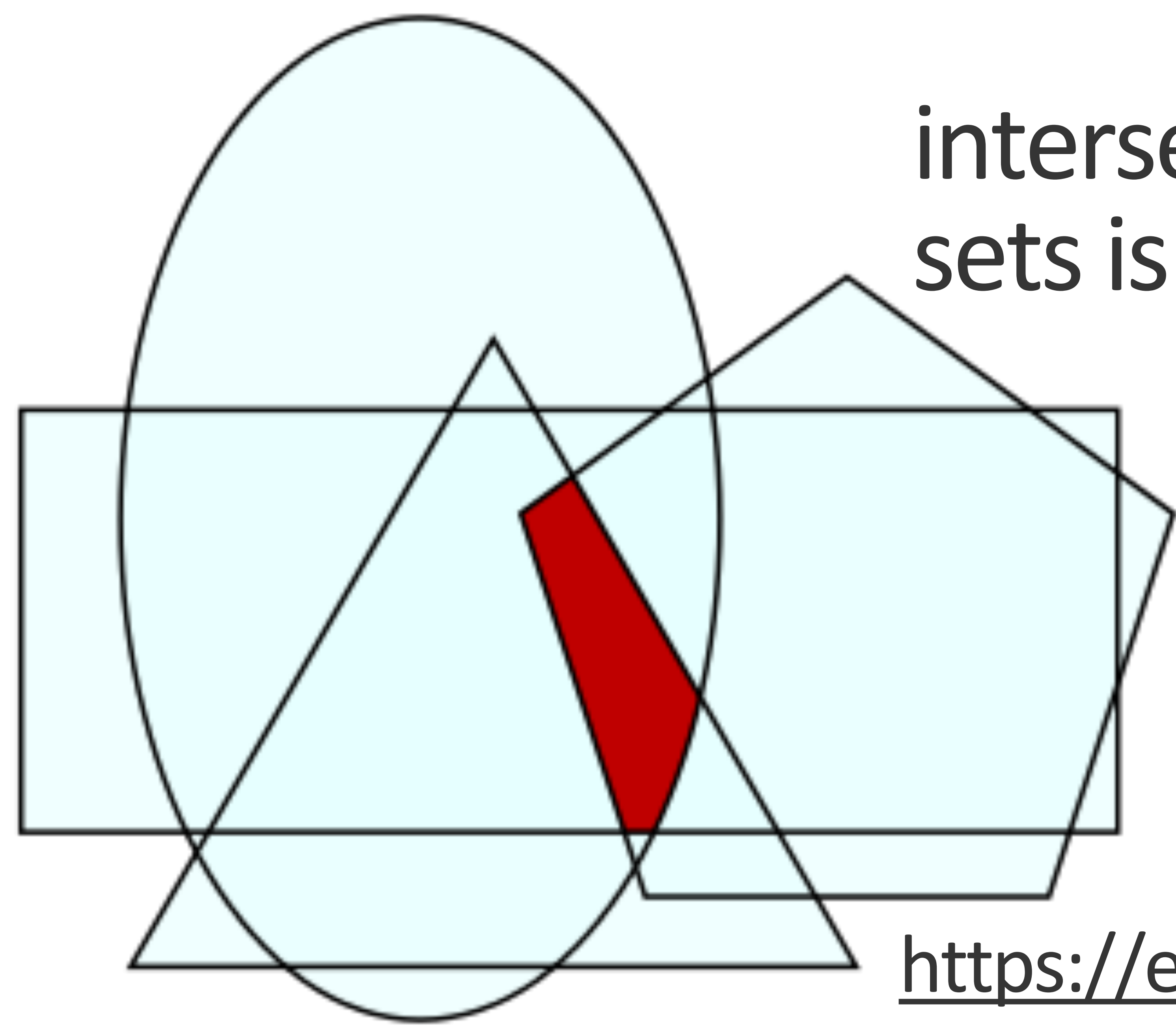
- single $\mathcal{C}$ (bounds) **[Zeev et al. (2006) and Bello and Raydan (2007)]**
- two sets **[Lelièvre and Oldenburg (2009), Baumstein (2013), Smithyman et al. (2015), Esser et al. (2015ab, 2016ab), B. Peters and Herrmann (2017)]**

$\mathbf{m}$

Convex sets : some properties

- line segment between every pair in the set, is in the set as well
- Euclidean projection onto a convex set is unique
- projection onto a convex set is a non-expansive operation





intersection of convex sets is also convex

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

SLIM

Prior information as convex sets

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

[Birgin et. al. (1999); Schmidt et. al. (2009); Schmidt et. al. (2012)]

projection based algorithms: SPG, PQN, projected Newton-type guarantee that $\mathbf{m}$ satisfies *all* constraints, *every* iteration.

Projection (Euclidean, minimum-distance projection):

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

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

Projection onto an intersection

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

Before, we used (parallel) black-box algorithms such as Dykstra's algorithm.

[Dykstra, 1983 ; Boyle & Dykstra, 1986 ; Censor, 2006; Bauschke & Koch, 2015]

one projection onto each set separately per iteration

Algorithm 1 Dykstra.

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

For $k = 0, 1, \ldots$

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

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

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

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

End

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 $k = 0, 1, \ldots$

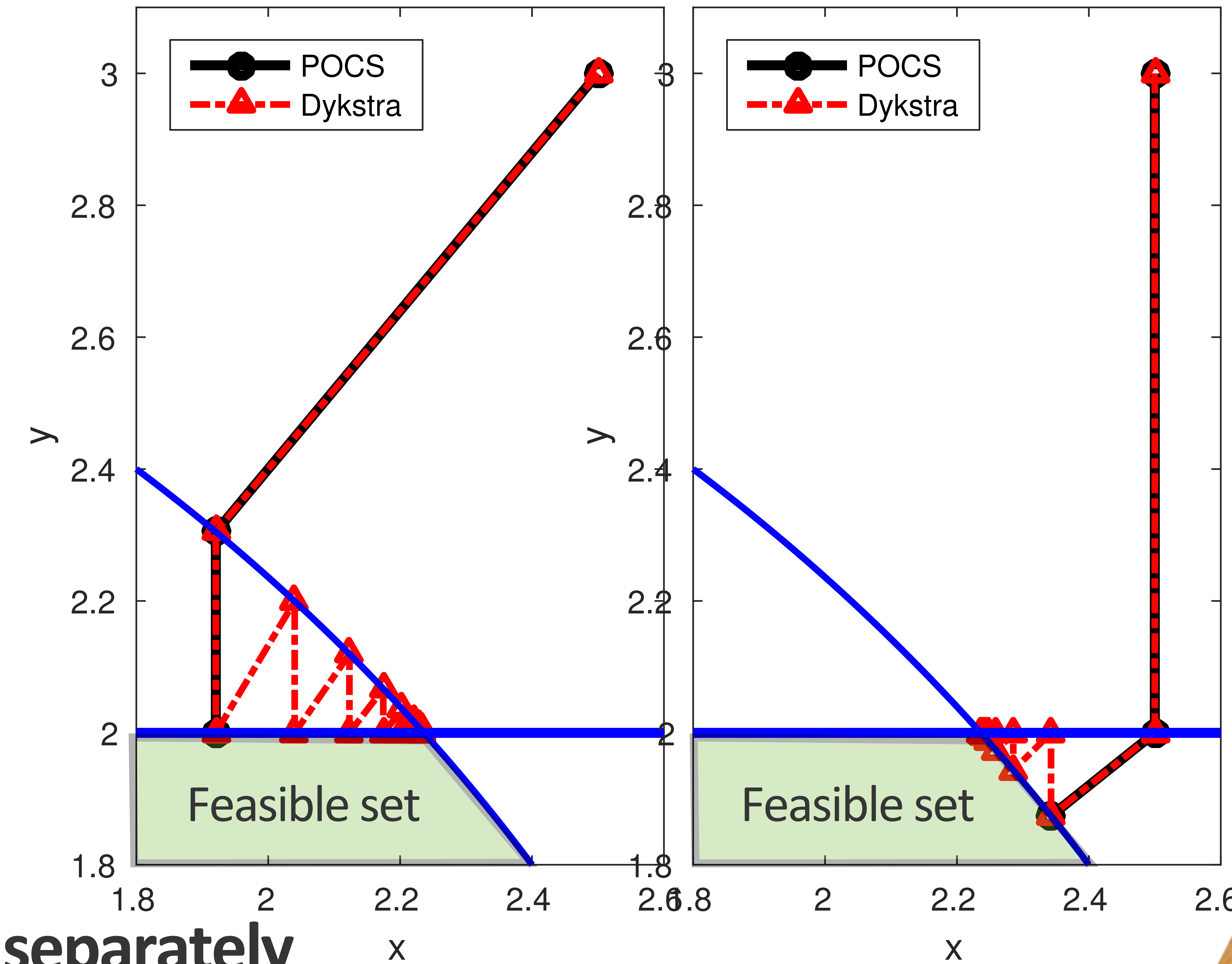
$\longrightarrow$ $y_k = \mathscr{P}_{C_1}(x_k + p_k)$

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

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

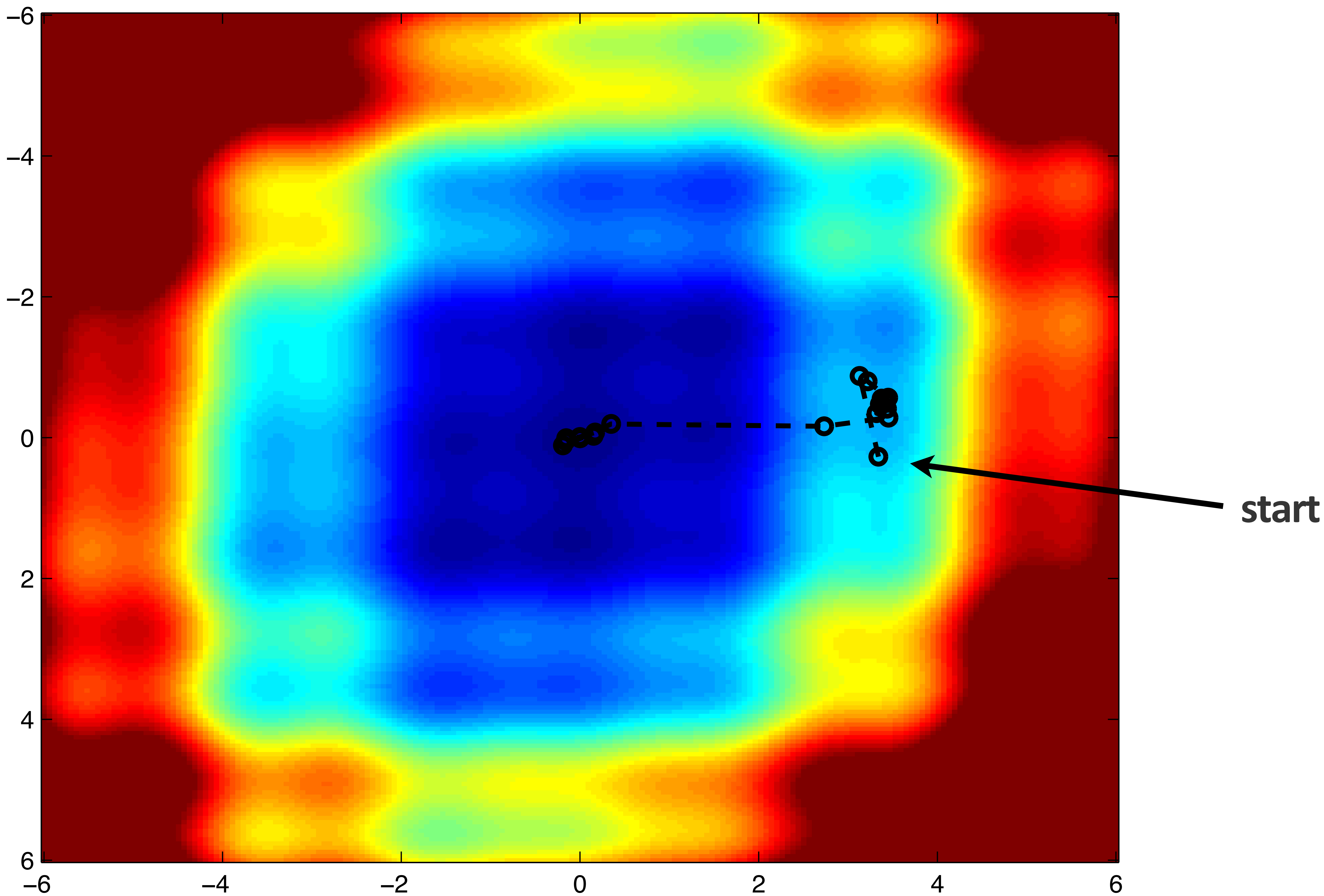
$q_{k+1} = y_k + q_k - x_{k+1}$

End



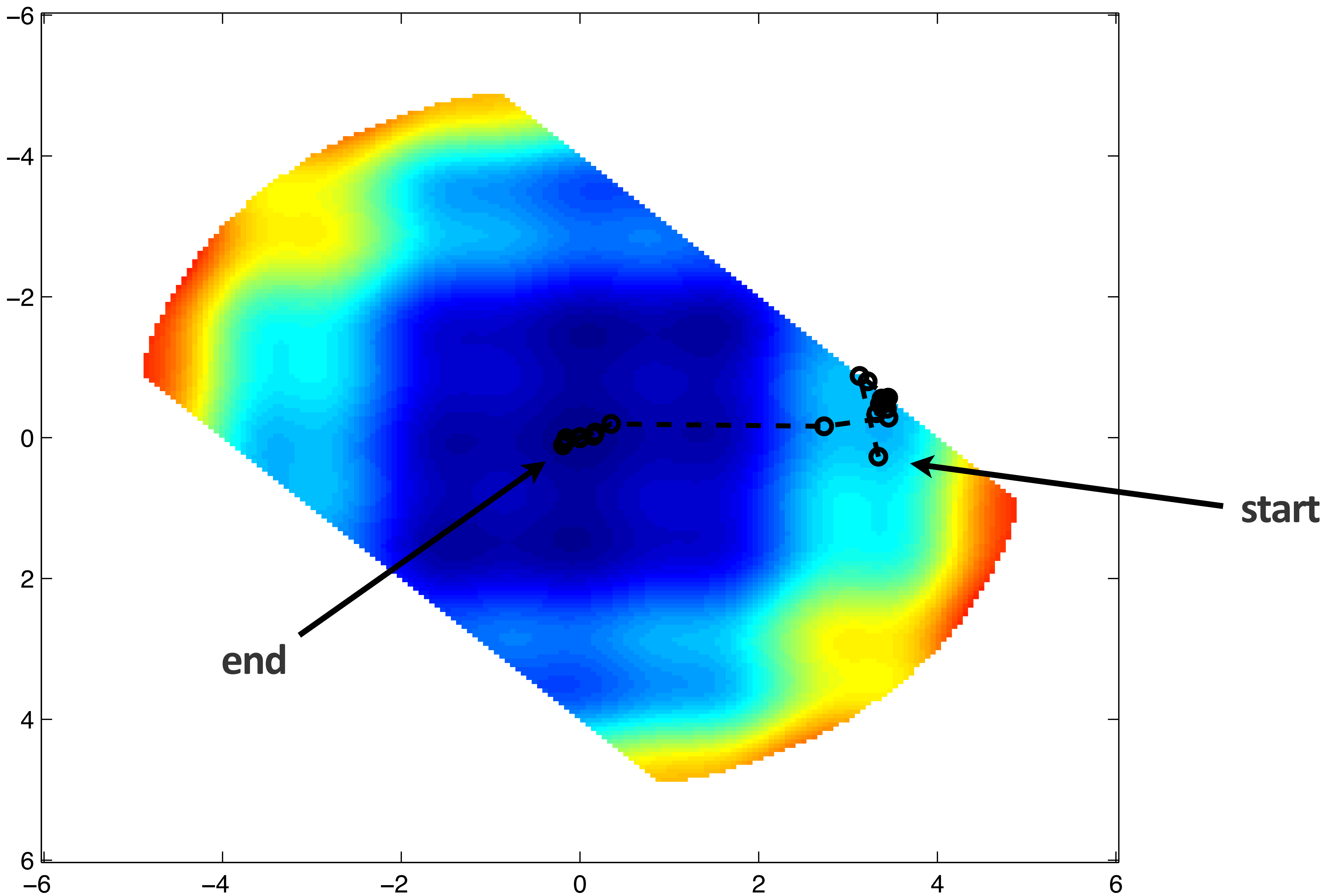
Only needs projections onto each set separately

Projected gradient



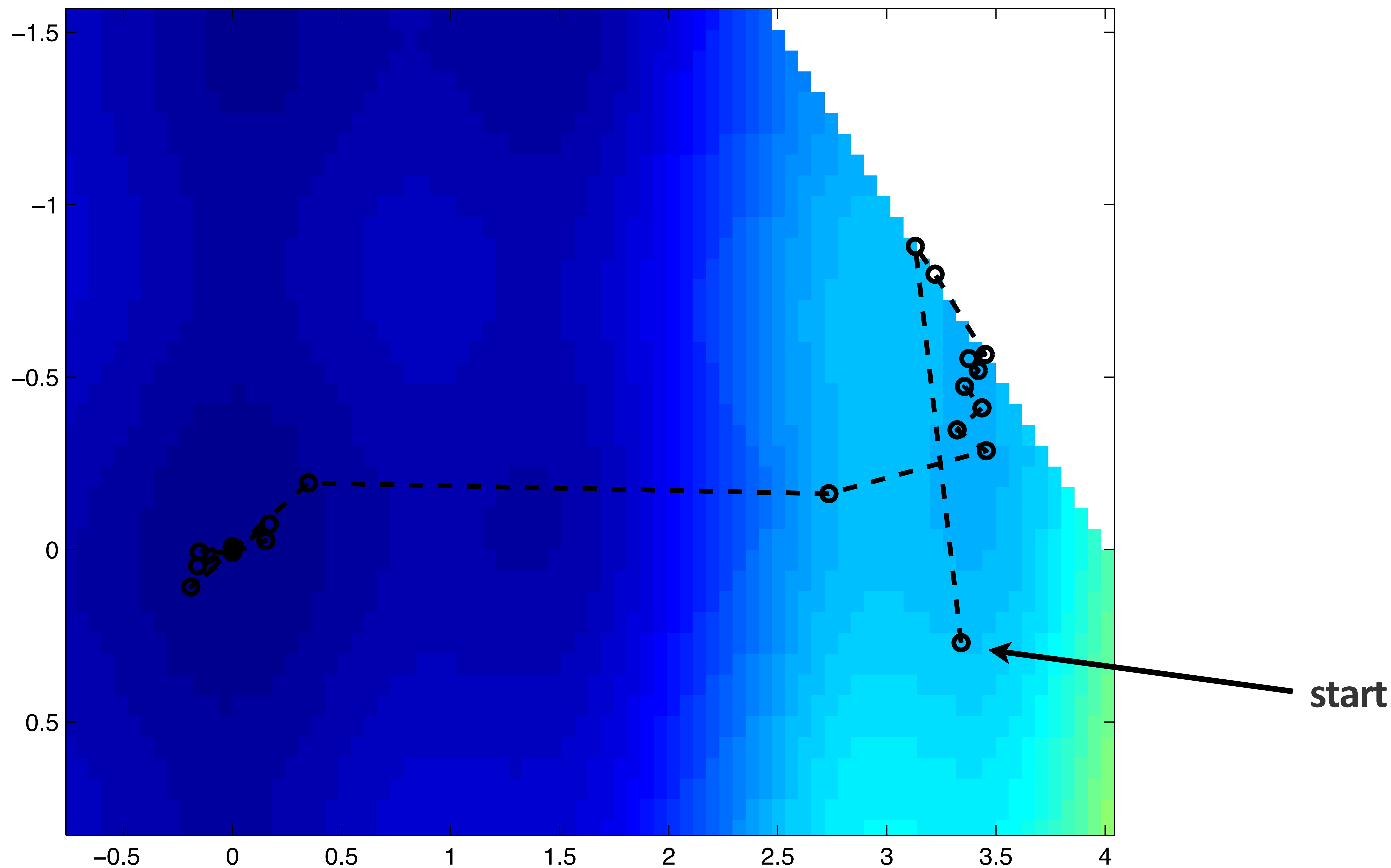
full non-convex function

Projected gradient



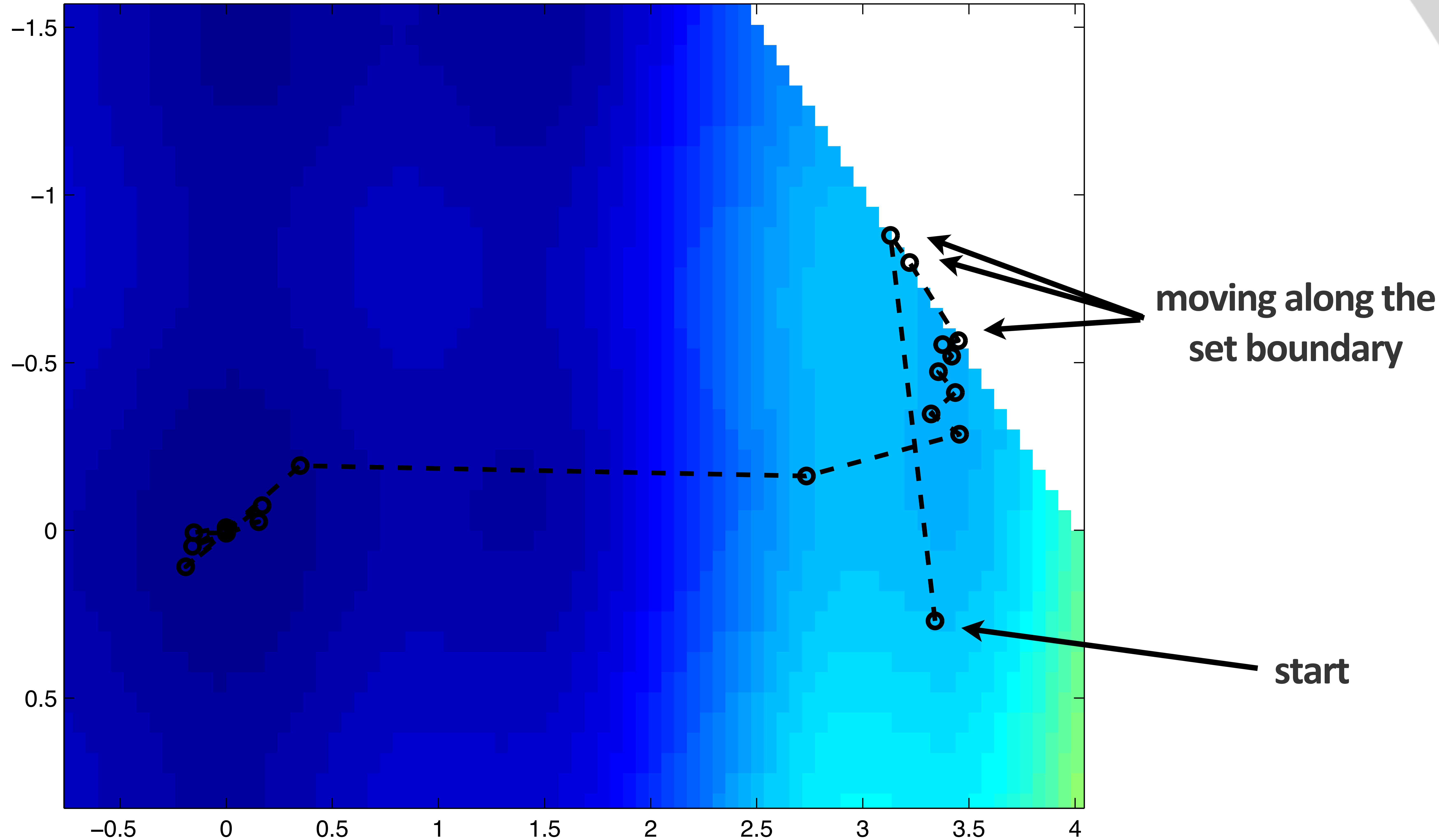
effective domain non-convex function

Projected gradient



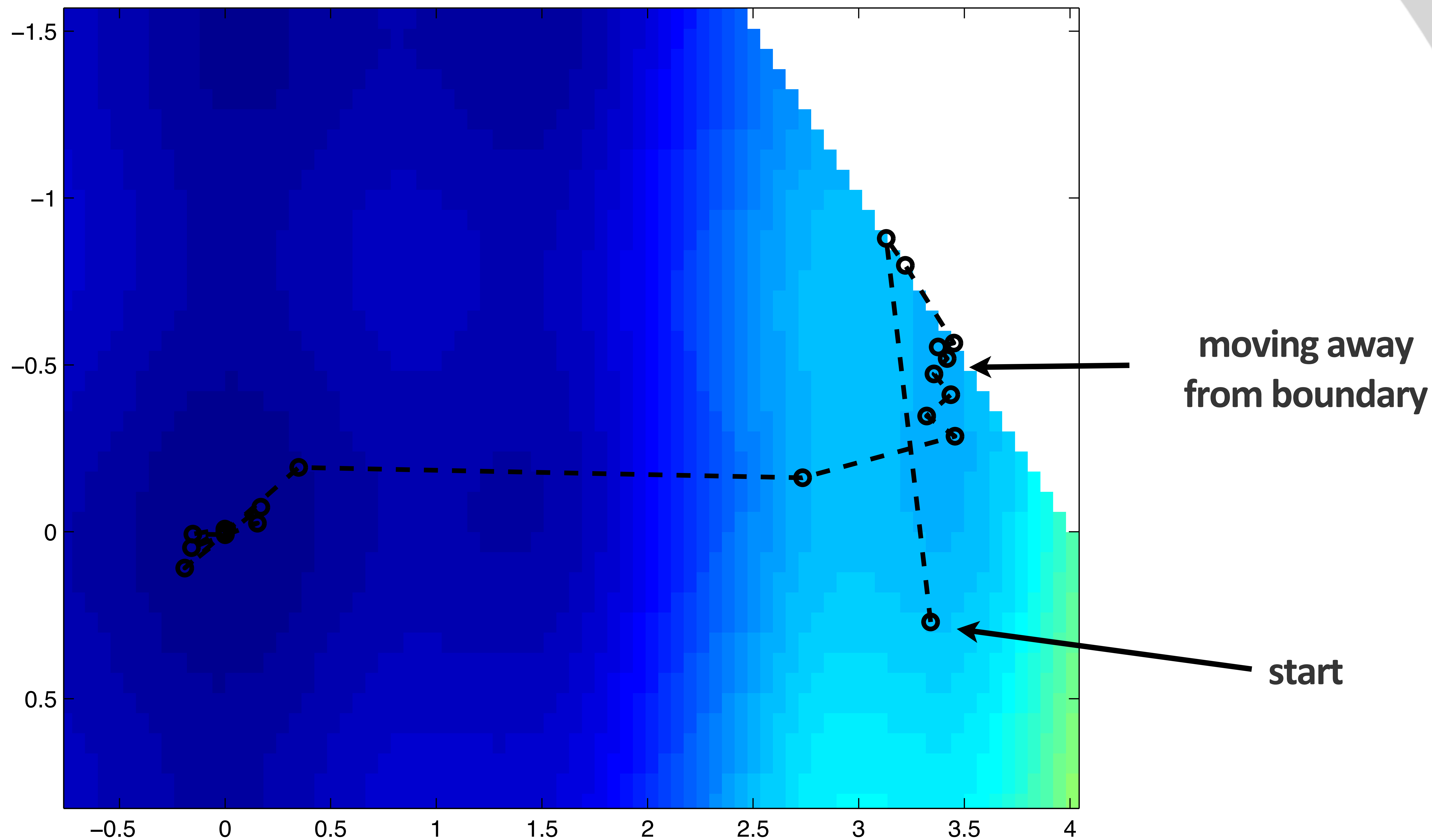
effective domain non-convex function zoomed in

Projected gradient



effective domain non-convex function zoomed in

Projected gradient



effective domain non-convex function zoomed in

Projection onto an intersection

Dykstra **Pro**:

- Simple and fast if projections are known in closed-form.

Dykstra **Con**:

- Uses another iterative algorithm for other projections
- Nested strategy requires two sets of stopping criteria.
- Does not take similarity between sets into account.

Algorithm 1 Dykstra.

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

For $k = 0, 1, \ldots$

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

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

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

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

End

Similarity between sets

limited number of discontinuities (lateral): $\{\mathbf{m} \mid \mathbf{card}(D_x \mathbf{m}) \leq k\}$

limited magnitude of discontinuities (lateral): $\{\mathbf{m} \mid \mathbf{l} \leq D_x \mathbf{m} \leq \mathbf{u}\}$

→ both sets have same transform-domain operator

anisotropic total-variation: $\{\mathbf{m} \mid \| \begin{pmatrix} D_z^T & D_x^T \end{pmatrix}^T \mathbf{m} \|_1 \leq \sigma\}$

limited number of discontinuities (lateral): $\{\mathbf{m} \mid \mathbf{card}(D_x \mathbf{m}) \leq k\}$

→ transform-domain operators have overlapping sparsity-pattern

→ mat-vec product at same cost

SLIM

New algorithm (1)

Goals:

Construct a single algorithm to project onto an intersection

- one instead of two sets of stopping criteria
- exploit similarity between sets
- use parallel resources

Merge ideas from SALSA/SDMM and ARADMM

- recast as known algorithm for known problem → convergence guarantees
- automatic (acceleration) parameter selection

[Afonso et. al., 2011], [Combettes & Pesquet, 2011 ; Kitic et. al. 2016] , [Xu et. al. ,2016a ; Xu et. al. ,2017]

New algorithm (2)

Reformulate projection onto an intersection:

$$\mathcal{P}_{\mathcal{C}}(\mathbf{m}) = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p-1} \iota_{\mathcal{C}_i}(A_i\mathbf{x})$$

Introduce new variables and couple w/ linear equality constraints:

$$\min_{\mathbf{x},\mathbf{y}_i} \frac{1}{2}\|\mathbf{x}-\mathbf{m}\|_2^2 + \sum_{i=1}^{p-1} \iota_{\mathcal{C}_i}(\mathbf{y}_i) \quad \text{s.t.} \quad A_i\mathbf{x} = \mathbf{y}_i$$

New algorithm (3)

Define matrix and vectors:

$$\tilde{A} \equiv \begin{pmatrix} A_1 \\ \vdots \\ A_p = I_N \end{pmatrix}, \quad \tilde{\mathbf{y}} \equiv \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_p \end{pmatrix}, \quad \tilde{\mathbf{v}} \equiv \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_p \end{pmatrix}$$

Define function:

$$\tilde{f}(\tilde{\mathbf{y}}) \equiv f(\mathbf{y}_p) + \sum_{i=1}^{p-1} \iota_{\mathcal{C}_i}(\mathbf{y}_i)$$

Final problem formulation:

$$\min_{\mathbf{x},\tilde{\mathbf{y}}} \tilde{f}(\tilde{\mathbf{y}}) \quad \text{s.t.} \quad \tilde{A}\mathbf{x} = \tilde{\mathbf{y}}$$

Equivalent to ADMM structure:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \ \text{ s.t. } \ A\mathbf{x} + B\mathbf{y} = \mathbf{c}$$

New algorithm (4)

ADMM is based on augmented Lagrangian: (separable in our case)

$$L_{\rho_1,\ldots,\rho_p}(\mathbf{x},\mathbf{y}_1,\ldots,\mathbf{y}_p,\mathbf{v}_1,\ldots,\mathbf{v}_p) =$$
$$\sum_{i=1}^{p}\left[f_i(\mathbf{y}_i) + \mathbf{v}_i^T(\mathbf{y}_i - A_i\mathbf{x}) + \frac{\rho_i}{2}\|\mathbf{y}_i - A_i\mathbf{x}\|_2^2\right]$$

ADMM iterations:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x},\mathbf{y}^k,\mathbf{v}^k)$$
$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} L_\rho(\mathbf{x}^{k+1},\mathbf{y},\mathbf{v}^k)$$
$$\mathbf{v}^{k+1} = \mathbf{v}^k + \rho(A\mathbf{x}^{k+1} - \mathbf{y}^{k+1})$$

New algorithm (5)

Iterations for our problem: (equivalent to SDMM + over/under relaxation)

$$\mathbf{x}^{k+1} = \Big(\sum_{i=1}^{p-1}[\rho_i A_i^T A_i] + \rho_p I_n\Big)^{-1} \sum_{i=1}^{p} \Big[A_i^T(\rho_i^k \mathbf{y}_i^k + \mathbf{v}_i^k)\Big]$$

$$\bar{\mathbf{x}}_i^{k+1} = \gamma_i^k A_i \mathbf{x}_i^{k+1} + (1-\gamma_i^k)\mathbf{y}_i^k$$

$$\mathbf{y}_i^{k+1} \in \mathbf{prox}_{f_i,\rho_i}(\bar{\mathbf{x}}_i^{k+1} - \frac{\mathbf{v}_i^k}{\rho_i^k})$$

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \rho_i^k(\mathbf{y}_i^{k+1} - \bar{\mathbf{x}}_i^{k+1})$$

New algorithm (6)

- Converges for $\rho_i > 0$ and $\gamma_i \in (0, 2)$

- Automatic updating of ρ_i and γ_i , based on Barzilai-Borwein **[Xu et. al. ,2016a ; Xu et. al. ,2017]**
- Uses equivalence between ADMM for

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \text{ s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{c}$$

  and Douglas-Rachford splitting on its dual problem
- Fewer iterations **[Xu et. al. ,2016a ; Xu et. al. ,2017]**
- Strong empirical performance on non-convex problems **[Xu et. al. ,2016b]**

New algorithm (7)

Iterations for our problem: (equivalent to SDMM + over/under relaxation)

$$\mathbf{x}^{k+1} = \Big(\sum_{i=1}^{p-1} [\rho_i A_i^T A_i] + \rho_p I_n \Big)^{-1} \sum_{i=1}^{p} \Big[A_i^T (\rho_i^k \mathbf{y}_i^k + \mathbf{v}_i^k) \Big] \longrightarrow \text{warm-start CG}$$

$$\bar{\mathbf{x}}_i^{k+1} = \gamma_i^k A_i \mathbf{x}_i^{k+1} + (1 - \gamma_i^k) \mathbf{y}_i^k$$

$$\mathbf{y}_i^{k+1} \in \mathbf{prox}_{f_i, \rho_i} (\bar{\mathbf{x}}_i^{k+1} - \frac{\mathbf{v}_i^k}{\rho_i^k}) \longrightarrow$$

simple projection onto set:
norm-ball/bounds/cardinality/rank
(all closed-form solutions)

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \rho_i^k (\mathbf{y}_i^{k+1} - \bar{\mathbf{x}}_i^{k+1})$$

New algorithm vs black-box approach

- Black-box version of the new algorithm can be derived as well
- Similar to parallel Dykstra
- Moves A from x-computation to y-computation

$$\mathbf{x}^{k+1} = \Big(\sum_{i=1}^{p-1} [\rho_i A_i^T A_i] + \rho_p I_n \Big)^{-1} \sum_{i=1}^{p} \Big[A_i^T (\rho_i^k \mathbf{y}_i^k + \mathbf{v}_i^k) \Big]$$

→ becomes average instead of linear system

$$\bar{\mathbf{x}}_i^{k+1} = \gamma_i^k A_i \mathbf{x}_i^{k+1} + (1 - \gamma_i^k) \mathbf{y}_i^k$$

$$\mathbf{y}_i^{k+1} \in \mathbf{prox}_{f_i, \rho_i} \Big(\bar{\mathbf{x}}_i^{k+1} - \frac{\mathbf{v}_i^k}{\rho_i^k} \Big)$$

→ becomes ‘difficult’ projection involving transform-domain operator (another iterative algorithm)

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \rho_i^k (\mathbf{y}_i^{k+1} - \bar{\mathbf{x}}_i^{k+1})$$

SLIM

Mixing column/row/fibre, matrix & tensors constraints

Consider prior knowledge: 5 main geological units

We expect max 4 large discontinuities in depth direction:

- 1 matrix based constraint:

$$\{\mathbf{m} \mid \mathbf{card}((D_z \otimes I_x)\mathbf{m}) \leq k\}$$
$$k = 4 \times N_{\text{gridpoints(x)}}$$

or

- $N_{\text{gridpoints(x)}}$ vector based constraints

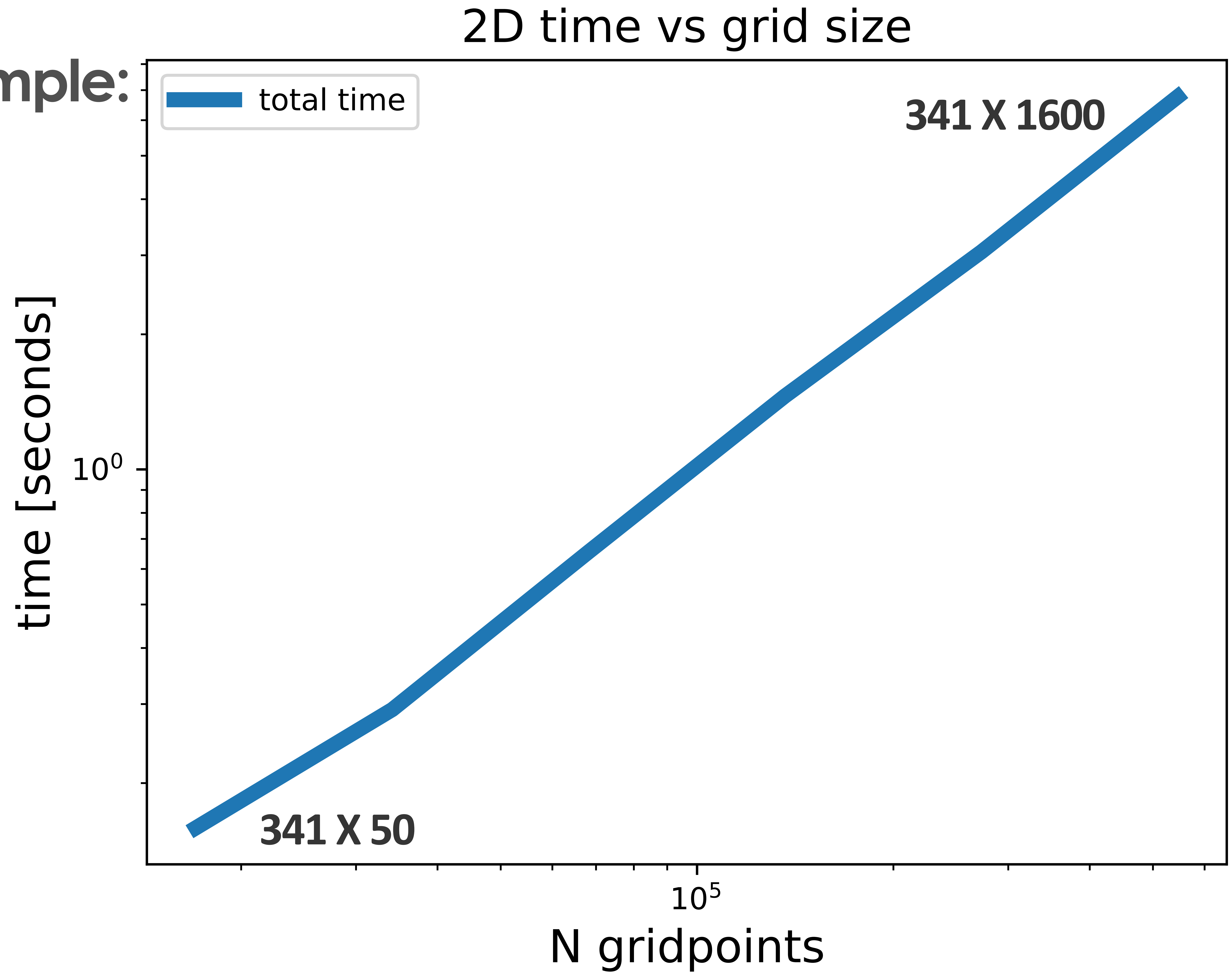
$$\{\mathbf{m} \mid \mathbf{card}(D_z R_i \mathbf{m}) \leq k\}$$
$$k = 4$$

- Software can use both simultaneously, both offer complementary information.
- Restriction matrix R_i drops out, does not occur in computations.

Timing 2D (serial)

Same constraints as in example:

- bounds on lateral gradient
- approximate vertical monotonicity
- bound constraints

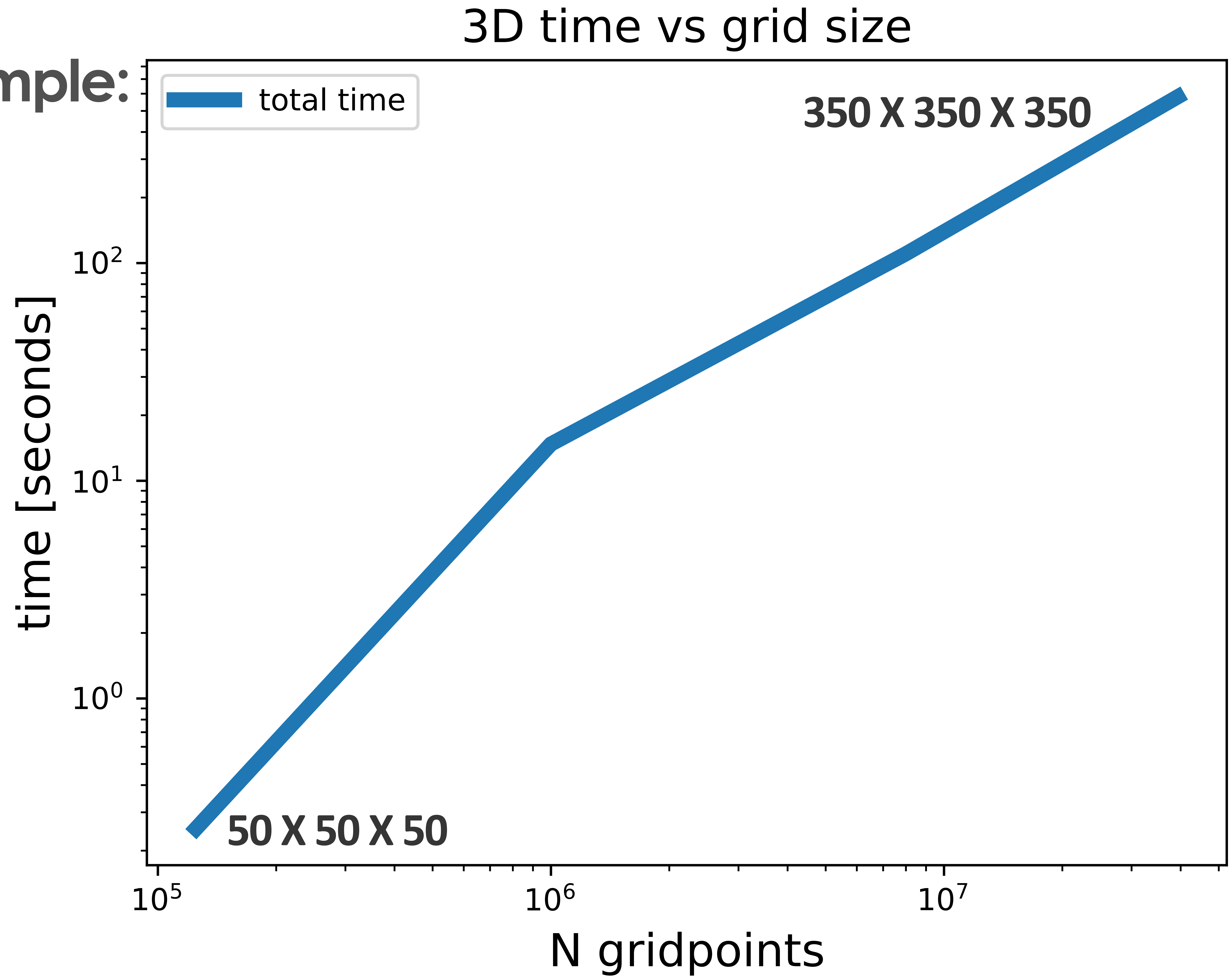


Timing 3D (serial)

Same constraints as in example:

- bounds on lateral gradient
- approximate vertical monotonicity
- bound constraints

- use domain-decomposition and/or multi-grid for larger domains



Software design (1)

Each set has two elementary components:

- transform-domain operator A
- sub-problem projection (closed-form) (norm-ball, cardinality, bounds, ...)

For example:

$\mathcal{C} \equiv \{\mathbf{m} \mid \mathbf{card}(A\mathbf{m}) \leq k\} \longrightarrow \mathcal{P}_{\mathbf{card}} =$ keep largest k elements & $A = A$

$\mathcal{C} \equiv \{\mathbf{m} \mid \|A\mathbf{m}\|_1 \leq \sigma\} \longrightarrow \mathcal{P}_{\|\cdot\|}$ & $A = A$

$\mathcal{C} \equiv \{\mathbf{m}_i \mid \mathbf{b}_i^l \leq \mathbf{m}_i \leq \mathbf{b}_i^u\} \longrightarrow A = I$ & $\mathcal{P}_{\mathcal{C}}(\mathbf{m}_i) = \mathrm{median}\{\mathbf{b}_i^l, \mathbf{m}_i, \mathbf{b}_i^u\}$

Software design (2)

Algorithm input:

- pairs of (transform-domain operator, sub-problem projection)$(A_i, \mathcal{P}_{\mathcal{C}_i})$
- point to project onto the intersection: $\mathbf{m}$

```
FL=32 #single precision (64 for double)

constraint=Dict()

#bound constraints
constraint["use_bounds"] = true
constraint["m_min"]      = 1450
constraint["m_max"]      = 5000

#rank constraints
constraint["use_rank"] = true
constraint["max_rank"] = 3

 #cardinality on derivatives (column or row wise)
 constraint["use_TD_card_fibre_x"]       = true
 constraint["card_fibre_x"]              = 3
 constraint["TD_card_fibre_x_operator"] = "D_x"

 #cardinality on derivatives (matrix based)
 constraint["use_TD_card_1"]       = true
 constraint["card_1"]              = round(Integer,3*0.33*n[1])
 constraint["TD_card_operator_1"] = "D_x"
```

```
#script that sets up transform-domain operators and sub-problem projectors
(P,P_sub,TD_OP,TD_Prop,AtA) = setup_constraints_2D(constraint,model,FL);

options_PARSDMM=PARSDMM_options() #get default solver options

#define function or function handle, input: model vector -> output: projected vector
function ProjectionIntersection(x)
  (x,log_PARSDMM)=compute_projection_intersection_PARSDMM(x,ini_guess,AtA,TD_OP,TD_Prop,
                                                          P_sub,constraint,options_PARSDMM)
  return x
end

#data misfit is a function / function(handle) with:
# input: model vector (m)
# output: data-misfit value (f) and gradient vector (g)
# (f,g) = data_misfit(m)

#FWI with spectral projected gradient algorithm (SPG)
(x, fsave, funEvals) = SPG(data_misfit, m0, ProjectionIntersection, SPG_options)
```

Conclusions

- add arbitrarily many constraints to existing FWI algorithms
- simpler, faster algorithms, also for non-convex sets (empirically)
- Julia implementation will be on SLIM git soon
- applies to other inverse problems as well

Acknowledgements

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

SINBAD