Algorithms and Julia software for constrained FWI

Bas Peters
SINBAD FALL 2017
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
Constraints

Currently implemented:

• bounds
• nuclear norm, rank
• $\ell_1$ - based sparsity promotion total-variation/transform-domain sparsity
• cardinality ($\ell_0$) - based total-variation transform-domain sparsity constraints
• slope constraints / transform-domain bounds
• Fourier-domain smoothness / subspace constraints
Transform-domain bounds / slope constraints

\[ C \equiv \{ m \mid l_j \leq (Am)_j \leq u_j \} \]

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

**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

Friday, October 6, 2017
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.

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):
  \[ \{ m \mid -\varepsilon_1 \leq ((I_z \otimes D_x)m)_j \leq +\varepsilon_2 \}\]
- approximate vertical monotonicity:
  \[ \{ m \mid -\varepsilon \leq ((D_z \otimes I_x)m)_j \leq +\infty \}\]

2nd cycle:
- use 1st cycle result as new starting model
- invert all data with bound constraints
With constraints, cycle 1

bounds & slope constraint, 3-4 Hz, iter = 1
bounds & slope → bounds only, 3–15 Hz, iter = 1
Problem formulation

\[
\min_m f(m) \quad \text{s.t.} \quad m \in \bigcap_{i=1}^{p} C_i
\]

differentiable data-misfit
intersection of constraint sets

Geophysical applications:
• single \( C \) (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)]
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

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

Intersection of convex sets is also convex
Prior information as convex sets

\[ \min_{\mathbf{m}} f(\mathbf{m}) \quad \text{s.t.} \quad \mathbf{m} \in \bigcap_{i=1}^{p} C_i \]

[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):

\[
P_C(\mathbf{m}) = \arg \min_{\mathbf{x}} \| \mathbf{x} - \mathbf{m} \|_2 \quad \text{s.t.} \quad \mathbf{x} \in C \quad P_C(\mathbf{m}) = P_C(P_C(\mathbf{m}))
\]
Projection onto an intersection

\[ P_C(m) = \arg \min_x \|x - m\|_2 \quad \text{s.t.} \quad x \in \bigcap_{i=1}^p C_i. \]

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

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

one projection onto each set separately per iteration

\[ x_0 = m, \quad p_0 = 0, \quad q_0 = 0 \]

For \( k = 0, 1, \ldots \)

\[ y_k = P_{C_1}(x_k + p_k) \]

\[ p_{k+1} = x_k + p_k - y_k \]

\[ x_{k+1} = P_{C_2}(y_k + q_k) \]

\[ q_{k+1} = y_k + q_k - x_{k+1} \]

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 \( k = 0, 1, \ldots \)

\[ y_k = \mathcal{P}_{C_1}(x_k + p_k) \]

\[ p_{k+1} = x_k + p_k - y_k \]

\[ x_{k+1} = \mathcal{P}_{C_2}(y_k + q_k) \]

\[ q_{k+1} = y_k + q_k - x_{k+1} \]

End

Only needs projections onto each set separately
Projected gradient

full non-convex function

start
Projected gradient

effective domain non-convex function
Projected gradient

Effective domain non-convex function zoomed in

start
Projected gradient

effective domain non-convex function zoomed in

moving along the set boundary

start

Friday, October 6, 2017
Projected gradient

effective domain non-convex function zoomed in

moving away from boundary

start
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.**

\[
\begin{align*}
x_0 &= m, \quad p_0 = 0, \quad q_0 = 0 \\
\text{For } k = 0, 1, \ldots \\
y_k &= \mathcal{P}_{C_1} (x_k + p_k) \\
p_{k+1} &= x_k + p_k - y_k \\
x_{k+1} &= \mathcal{P}_{C_2} (y_k + q_k) \\
q_{k+1} &= y_k + q_k - x_{k+1} \\
\end{align*}
\]

End
Similarity between sets

limited number of discontinuities (lateral): \( \{ m \ | \ \text{card}(D_x m) \leq k \} \)

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

\( \Rightarrow \) both sets have same transform-domain operator

anisotropic total-variation: \( \{ m \ | \ \| (D_z^T D_x^T)^T m \|_1 \leq \sigma \} \)

limited number of discontinuities (lateral): \( \{ m \ | \ \text{card}(D_x m) \leq k \} \)

\( \Rightarrow \) transform-domain operators have overlapping sparsity-pattern

\( \Rightarrow \) mat-vec product at same cost
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:

$$\mathcal{P}_C(m) = \arg\min_x \frac{1}{2} \|x - m\|_2^2 + \sum_{i=1}^{p-1} \iota_C(A_i x)$$

Introduce new variables and couple w/ linear equality constraints:

$$\min_{x,y_i} \frac{1}{2} \|x - m\|_2^2 + \sum_{i=1}^{p-1} \iota_C(y_i) \quad \text{s.t.} \quad A_i x = y_i$$
New algorithm (3)

Define matrix and vectors:

\[
\tilde{A} = \begin{pmatrix}
A_1 \\
\vdots \\
A_p = I_N
\end{pmatrix}, \quad \tilde{y} = \begin{pmatrix}
y_1 \\
\vdots \\
y_p
\end{pmatrix}, \quad \tilde{v} = \begin{pmatrix}
v_1 \\
\vdots \\
v_p
\end{pmatrix}
\]

Define function:

\[
\tilde{f}(\tilde{y}) = f(y_p) + \sum_{i=1}^{p-1} \lambda C_i(y_i)
\]

Final problem formulation:

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

Equivalent to ADMM structure:

\[
\min_{x,y} f(x) + g(y) \quad \text{s.t.} \quad Ax + By = c
\]
New algorithm (4)

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

\[
L_{\rho_1, \ldots, \rho_p}(x, y_1, \ldots, y_p, v_1, \ldots, v_p) = \\
\sum_{i=1}^{p} \left[ f_i(y_i) + v_i^T (y_i - A_i x) + \frac{\rho_i}{2} \|y_i - A_i x\|^2_2 \right]
\]

ADMM iterations: 

\[
x^{k+1} = \arg \min_x L_\rho(x, y^k, v^k)
\]

\[
y^{k+1} = \arg \min_y L_\rho(x^{k+1}, y, v^k)
\]

\[
v^{k+1} = v^k + \rho (A x^{k+1} - y^{k+1})
\]
New algorithm (5)

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

\[
x^{k+1} = \left( \sum_{i=1}^{p-1} [\rho_i A_i^T A_i] + \rho_p I_n \right)^{-1} \sum_{i=1}^{p} \left[ A_i^T (\rho_i^k y_i^k + v_i^k) \right]
\]

\[
x_i^{k+1} = \gamma_i^k A_i x_i^{k+1} + (1 - \gamma_i^k) y_i^k
\]

\[
y_i^{k+1} \in \text{prox}_{f_i, \rho_i} \left( \bar{x}_i^{k+1} - \frac{v_i^k}{\rho_i^k} \right)
\]

\[
v_i^{k+1} = v_i^k + \rho_i^k (y_i^{k+1} - \bar{x}_i^{k+1})
\]
New algorithm (6)

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

- Automatic updating of $\rho_i$ and $\gamma_i$, based on Barzilai-Borwein [Xu et. al., 2016a; Xu et. al., 2017]

- Uses equivalence between ADMM for

$$\min_{x,y} f(x) + g(y) \text{ s.t. } Ax + By = c$$

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)

\[
x^{k+1} = \left( \sum_{i=1}^{p-1} [\rho_i A_i^T A_i] + \rho_p I \right)^{-1} \sum_{i=1}^{p} A_i^T (\rho_i^k y_i^k + v_i^k)
\]

\[
x_i^{k+1} = \gamma_i^k A_i x_i^{k+1} + (1 - \gamma_i^k) y_i^k
\]

\[
y_i^{k+1} \in \text{prox}_{f_i, \rho_i} \left( \bar{x}_i^{k+1} - \frac{v_i^k}{\rho_i^k} \right)
\]

\[
v_i^{k+1} = v_i^k + \rho_i^k (y_i^{k+1} - \bar{x}_i^{k+1})
\]
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

\[ x^{k+1} = \left( \sum_{i=1}^{p-1} [\rho_i A_i^T A_i] + \rho_p I_n \right)^{-1} \sum_{i=1}^{p} \left[ A_i^T (\rho_i^{-1} y_i^k + v_i^k) \right] \]

becomes average instead of linear system

\[ \bar{x}_i^{k+1} = \gamma_i^k A_i x_i^{k+1} + (1 - \gamma_i^k) y_i^k \]

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

\[ y_i^{k+1} \in \text{prox}_{f_i, \rho_i} (\bar{x}_i^{k+1} - \frac{v_i^k}{\rho_i^k}) \]

\[ v_i^{k+1} = v_i^k + \rho_i^k (y_i^{k+1} - \bar{x}_i^{k+1}) \]
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:

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

or

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

\[ \{ \mathbf{m} \mid \text{card}(D_z R_i \mathbf{m}) \leq k \}\]
\[ k = 4 \]

• 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

![3D time vs grid size graph](image)
Software design (1)

Each set has two elementary components:
- transform-domain operator $A$
- sub-problem projection (closed-form) (norm-ball, cardinality, bounds, ...)

For example:

$C \equiv \{m \mid \text{card}(Am) \leq k\} \quad \Rightarrow \quad \mathcal{P}_{\text{card}} = \quad \text{keep largest } k \text{ elements } \& \quad A = A$

$C \equiv \{m \mid \|Am\|_1 \leq \sigma\} \quad \Rightarrow \quad \mathcal{P}_{\|\cdot\|} \quad \& \quad A = A$

$C \equiv \{m_i \mid b^l_i \leq m_i \leq b^u_i\} \quad \Rightarrow \quad A = I \quad \& \quad \mathcal{P}_C(m_i) = \text{median}\{b^l_i, m_i, b^u_i\}$
Software design (2)

Algorithm input:

- pairs of (transform-domain operator, sub-problem projection) \((A_i, P_{C_i})\)
- point to project onto the intersection: \(m\)
3  FL=32 # single precision (64 for double)

4  constraint=Dict()

7  # bound constraints
8  constraint["use_bounds"] = true
9  constraint["m_min"] = 1450
10 constraint["m_max"] = 5000

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

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

21  # cardinality on derivatives (matrix based)
22  constraint["use_TD_card_1"] = true
23  constraint["card_1"] = round(Integer,3*0.33*n[1])
24  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_PARSDDM=PARSDDM_options() #get default solver options

#define function or function handle, input: model vector -> output: projected vector
function ProjectionIntersection(x)
  (x,log_PARSDDM)=compute_projection_intersection_PARSDDM(x,ini_guess,AtA,TD_OP,TD_Prop, P_sub.constraint.options_PARSDDM)
  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.