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

Bas Peters
Motivation – noisy data

- True velocity model
- Initial velocity model
- Only bound constraints
Motivation – bad start model / missing low freq.

- True velocity model
- Initial velocity model
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
- $\ell_1$ - 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
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):

$$\mathcal{P}_C(m) = \arg \min_x \|x - m\|_2 \quad \text{s.t.} \quad x \in C_1 \bigcap C_2$$

Important property:

$$\mathcal{P}_C(m) = \mathcal{P}_C(\mathcal{P}_C(m))$$
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

\[ C \equiv \{ m \mid \|Am\|_1 \leq \sigma \} \]

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: \( S \equiv \{ m | \text{card}(Am) \leq k \} \). \( k \): 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.

\[ S \equiv \{ m \mid \text{card}(Am) \leq k \} \]

Convex 1-norm:
requires estimate of the **number** of major interfaces and the **magnitude** of the jumps

\[ C \equiv \{ m \mid \|Am\|_1 \leq \sigma \} \]
Rank constraints

non-convex set: $S_1 \equiv \{M_r \mid M_r = \sum_{j=1}^{r} \lambda_j u_j v_j^*\}$

Simplest form of the projector: SVD, $r<k$

$P_{S_1}(M) = \sum_{j=1}^{r} \lambda_j u_j v_j^*$, with $M = \sum_{j=1}^{k} \lambda_j u_j v_j^*$.

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:

\[ C \equiv \{ m \mid b^l \leq Am \leq b^u \} \]

Not clear how this could help in Wavelet, Curvelet or Fourier-domain. Useful in discrete-gradient domain:

\[
A = I_n \otimes D_z \quad D_z = \frac{1}{h_z} \begin{pmatrix}
-1 & 1 \\
-1 & 1 & \ddots & \ddots \\
& \ddots & -1 & 1
\end{pmatrix}
\]
Transform-domain bounds / slope constraints

\[ C \equiv \{ m_i \mid b^l_i \leq A m_i \leq b^u_i \} \quad \text{with} \quad A = I_n \otimes D_z \]

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
-\textit{induces} monotonicity

limited increase and limited decrease
-\textit{induces} vertical smoothness
-\textit{still allows} small velocity jumps
Design principles

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

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
Nested optimization strategy

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

1. Algorithm for nonconvex (smooth + nonsmooth) optimization

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

2. Algorithm computing the projection onto an intersection

\[
\mathcal{P}_C(\mathbf{m}) = \arg \min_{\mathbf{x}} \|\mathbf{x} - \mathbf{m}\|_2 \quad \text{s.t.} \quad \mathbf{x} \in \bigcap_{i=1}^{p} C_i.
\]

3. Projection onto each set separately

\[
\mathcal{P}_{C_i}(\mathbf{m}) = \arg \min_{\mathbf{x}} \|\mathbf{x} - \mathbf{m}\|_2 \quad \text{s.t.} \quad \mathbf{x} \in C_i.
\]
Projected gradient

full non-convex function

start
Projected gradient

effective domain non-convex function

start

end
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
Projected gradient

effective domain non-convex function zoomed in

moving away from boundary

start
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!

Figure 1: The trajectory of Dykstra’s algorithm for a toy example with constraints \( y \leq 2 \) and \( x \leq 2 \).

Iterates 5, 6 and 7 coincide, the algorithm converged to the point closest to point number 1 and satisfying both constraints. Note that the projection onto convex sets (POCS) algorithm would converge at point number 3 and is clearly unsuitable for this type of projection problem.

While the gradient-projection algorithm is a solid approach, it can also be relatively slowly converging. A potentially much faster algorithm is the class of quasi-Newton methods, which iteratively try to approximate the Hessian by using just gradient and function value information. However, in general it is not possible to just project quasi-Newton steps onto a convex set, just as in the gradient-projection algorithm. The use of second-order information may cause a projected step to point in the opposite direction and in general does not solve problem 5, but converges to a solution which does not correspond to the problem. There are slightly more complicated algorithms which properly implement a projected quasi-Newton algorithm. We select the projected-quasi-Newton (PQN) algorithm by Schmidt et al. (2009). This algorithm finds a search direction which is in the intersection of the convex sets using the spectral projected gradient algorithm (SPG) as a subproblem of an L-BFGS-like algorithm. This algorithm needs the objective function value, \( f(m) \), its gradient and an algorithm to solve the projection problem 6. This algorithm has very similar computational cost as the standard LBFGS algorithm, because the projections are cheap to compute. See Schmidt et al. (2009) for some examples of strong empirical performance of PQN versus projected-gradient for some non-geophysical examples.
Projection computation

\[ P_C(m) = \arg \min_x \frac{1}{2} \|x - m\|_2^2 \quad \text{s.t} \quad x \in C \]

If no closed form solution is available: reformulate and solve using ADMM. This works for all transform-domain norm/cardinality/bounds

\[ P_C(m) = \arg \min_x \frac{1}{2} \|x - m\|_2^2 \quad \text{s.t} \quad x \in C \]

\[ = \arg \min_x \frac{1}{2} \|x - m\|_2^2 \quad \text{s.t} \quad \|Ax\| \leq \sigma \]

\[ = \arg \min_x \frac{1}{2} \|x - m\|_2^2 \quad \text{s.t} \quad \|z\| \leq \sigma , \ Ax = z \]
Projection computation

\[ P_C(m) = \arg \min_x \frac{1}{2} \| x - m \|^2 \quad \text{s.t} \quad x \in C \]

If no closed form solution is available: reformulate and solve using ADMM.
This works for all transform-domain norm/cardinality/bounds

\[ P_C(m) = \arg \min_x \frac{1}{2} \| x - m \|^2 \quad \text{s.t} \quad x \in C \]

\[ = \arg \min_x \frac{1}{2} \| x - m \|^2 \quad \text{s.t} \quad \text{card}(Ax) \leq \sigma \]

\[ = \arg \min_x \frac{1}{2} \| x - m \|^2 \quad \text{s.t} \quad \text{card}(z) \leq \sigma , Ax = z \]
Projection computation

\[ P_C(m) = \arg \min_x \frac{1}{2} \| x - m \|_2^2 \quad \text{s.t} \quad x \in C \]

If no closed form solution is available: reformulate and solve using ADMM. This works for all transform-domain norm/cardinality/bounds

\[ P_C(m) = \arg \min_x \frac{1}{2} \| x - m \|_2^2 \quad \text{s.t} \quad x \in C \]

\[ = \arg \min_x \frac{1}{2} \| x - m \|_2^2 \quad \text{s.t} \quad b^l \leq Ax \leq b^u \]

\[ = \arg \min_x \frac{1}{2} \| x - m \|_2^2 \quad \text{s.t} \quad b^l \leq z \leq b^u, \; Ax = z \]
Projection computation

\[ P_c(m) = \arg \min_x \frac{1}{2} \|x - m\|^2 \quad \text{s.t.} \quad \|z\| \leq \sigma, Ax = z \]

%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

User provided code for FWI

- gradient
- function value

Optimization algorithm which handles projections (projected & proximal algorithms)

Algorithm to compute projection onto intersection

Projector onto set 1
Projector onto set 2
Projector onto set p

vector
projected vector
User provides a code which computes function values and gradients:

```matlab
%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:

```matlab
[f,g]= data_misfit(m);
```
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 frequency and initial model.

% Obtain projectors onto each set separately

\[
[\text{Proj\_bound}, \text{Proj\_TV}, \text{Proj\_rank}] = \text{setup\_constraints}(\text{constraint}, \text{geometry}, m0, \text{frequencies});
\]

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

\[
\text{output} = \text{Proj\_TV}(\text{input})
\]
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.
Call optimization algorithm using the data-misfit & gradient function handle and the intersection projector.

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

\[ C \equiv \{ m | b^l \leq Am \leq b^u \} \]

In this example \( Am \) means

\[ b^l_i \leq \frac{m_{i+1,j} - m_{i,j}}{h_z} \leq \infty \]

In words:

velocity can increase with depth, unbounded.

velocity can only slowly decrease with depth. \( b^l_i = \) small negative number
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.
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
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

True velocity 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.


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

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.