Pros and Cons of Full- and Reduced-space Methods for Wavefield Reconstruction Inversion

Felix J. Herrmann & Bas Peters

SLIM
University of British Columbia

SIAM Conference on Mathematical and Computational Issues in the Geosciences
June 29, 2015

Released to public domain under Creative Commons license type BY (https://creativecommons.org/licenses/by/4.0).
Copyright (c) 2015 SLIM group @ The University of British Columbia.
Pros and Cons of Full- and Reduced-space Methods for Wavefield Reconstruction Inversion

Felix J. Herrmann & Bas Peters
Inversion result
Model update
Source wavelet comparison

![Amplitude Graph](image1)

- True Wavelet
- Estimated Wavelet

![Phase Graph](image2)

- True Wavelet
- Estimated Wavelet
PDE-constrained optimization

Use the ‘discretize-then-optimize’ framework:

\[
\min_{m,u} \frac{1}{2} \| Pu - d \|^2_2 \quad \text{s.t.} \quad H(m)u = q
\]

- \( H(m) \in \mathbb{C}^{N \times N} \) discrete PDE
- \( m \in \mathbb{R}^N \) medium parameters
- \( P \in \mathbb{R}^{m \times N} \) selects field at receivers
- \( u \in \mathbb{C}^N \) field
- \( d \in \mathbb{C}^m \) observed data
- \( q \in \mathbb{C}^N \) source

[E. Haber & U.M. Ascher, 2001; G. Biros & O. Ghattas, 2005; Grote et. al., 2011]
\[
\min_{m,u} \frac{1}{2} \|Pu - d\|^2 \quad \text{s.t.} \quad H(m)u = q
\]

\[
\mathcal{L}(m, u, v) = \frac{1}{2} \|Pu - d\|^2 + v^*(H(m)u - q)
\]

**All-at-once full-space:** [E. Haber & U.M. Ascher, 2001; G. Biros & O. Ghattas, 2005; Grote et. al., 2011]

- update fields, medium parameters & multipliers simultaneously
- function value, gradient, Hessian evaluation is ~free & exact
- sparse Hessian
- requires storage of all fields & multipliers + working memory (gradients, Hessian & update direction)
- updates are computationally demanding
\[
\min_{m,u} \frac{1}{2} \| Pu - d \|_2^2 \quad \text{s.t.} \quad H(m)u = q
\]

\[
\mathcal{L}(m, u, v) = \frac{1}{2} \| Pu - d \|_2^2 + v^* (H(m)u - q)
\]

**Reduced space:** [Tarantola, ’84; Haber et al., ‘00; Epanomeritakis et al., ‘08]
- storage as low as two fields at a time
- highly nonlinear in \( m \) & non-convex
- expensive “exact” forward & adjoint solves for each iteration
- inexact when sub-problems are solved iteratively
- dense reduced-Hessian
- requires extra safeguards/accuracy control \[T.\, van\, Leeuwen\, &\, F.J.\, H\, ‘14\]
- reliance on accurate starting models to avoid cycle skipping
Example from [Peters et al. 2014]
\[
\min_m \frac{1}{2} \| PH(m)^{-1} q - d \|^2
\]
Few algorithms have quadratic-penalty form:
[R.E. Kleinman & P.M. van den Berg, ‘92; T. van Leeuwen & F.J.H, ‘13]

\[
\begin{align*}
\min_{m,u} \frac{1}{2} \| Pu - d \|_2^2 & \quad \text{s.t.} \quad H(m)u = q \\
\min_{m,u} \frac{1}{2} \| Pu - d \|_2^2 + \frac{\lambda^2}{2} \| H(m)u - q \|_2^2 & \\
& \text{eliminate field variables: solve } \nabla_u \phi(m, \bar{u}, \lambda) = 0 \\
& \text{[T. van Leeuwen & F.J.H. ’13;’15]}
\end{align*}
\]

Penalty method:
- no need to store all the fields \( u \)
- no adjoint solves
- sparse approximation of Gauss-Newton Hessian for small \( \lambda \)
- less non-linear in \( m \)
- need to solve data-augmented wave equation

\[
\min_m \frac{1}{2} \| P\bar{u} - d \|_2^2 + \frac{\lambda^2}{2} \| H(m)\bar{u} - q \|_2^2
\]
reduced quadratic-penalty
Reduced-space quadratic-penalty method – Wavefield Reconstruction Inversion (WRI)

Minimize:
\[ \bar{\phi}(m, \bar{u}, \lambda) = \frac{1}{2} \|P\bar{u} - d\|_2^2 + \frac{\lambda^2}{2} \|H(m)\bar{u} - q\|_2^2 \]

at every iteration
- compute
  \[ \bar{u} = \arg\min_u \left\| \begin{pmatrix} \lambda H(m) & P \\ \lambda q & d \end{pmatrix} u - \begin{pmatrix} \lambda q \\ d \end{pmatrix} \right\|_2 \]
- evaluate
  \[ \bar{\phi}(m, \bar{u}, \lambda) \& \nabla_m \bar{\phi}(m, \bar{u}, \lambda) \]
- update
  \[ m \]

+ trust-region / line-search

[T. van Leeuwen & F.J. Herrmann, 2013]
Example from [Peters et al. 2014]
\[
\min_m \frac{1}{2} \| PH(m)^{-1} q - d \|_2^2
\]

\[
\min_m \frac{1}{2} \| P\bar{u} - d \|_2^2 + \frac{\lambda^2}{2} \| H(m)\bar{u} - q \|_2^2 \quad \text{(small } \lambda \text{)}
\]
Reduced-space sub-problems

Solve \( u = H^{-1} q \) or \( \bar{u} = \arg \min_u \left\| \left( \begin{array}{c} \lambda H(m) \\ P \end{array} \right) u - \left( \begin{array}{c} \lambda q \\ d \end{array} \right) \right\|_2 \)

In 3D we have iterative & inexact solves:
- variable elimination & projection are assumed exact
- inaccuracy may cause problems in reduced-space methods
  - how accurate do we need to be?
  - can we change the accuracy per iteration?
- costs are dominated by accuracy & # of source experiments
- heuristic solutions proposed in frugal approaches
  [T. van Leeuwen & F.J. Herrmann, 2014]
Inexact PDE solves
– full-space vs reduced-space

**reduced-space:**
- error in objective function value
- error in gradient
- error in Hessian

**full-space:**
- objective function value always exact
- gradient always exact
- Hessian always exact

Quadratic-penalty full space methods

\[
\min_{\mathbf{m}, \mathbf{u}} \frac{1}{2} \| P \mathbf{u} - \mathbf{d} \|_2^2 + \frac{\lambda^2}{2} \| H(\mathbf{m}) \mathbf{u} - \mathbf{q} \|_2^2
\]

Newton's method:

\[
\begin{pmatrix}
P^* P + \lambda^2 H^* H & \nabla_{\mathbf{m}, \mathbf{u}}^2 \phi \\
\nabla_{\mathbf{u}, \mathbf{m}}^2 \phi & \lambda^2 G_{\mathbf{m}}^* G_{\mathbf{m}}
\end{pmatrix}
\begin{pmatrix}
\delta \mathbf{u} \\
\delta \mathbf{m}
\end{pmatrix}
= -\begin{pmatrix}
P^*(P\mathbf{u} - \mathbf{d}) + \lambda^2 H^*(H\mathbf{u} - \mathbf{q}) \\
\lambda^2 G_{\mathbf{m}}^*(H\mathbf{u} - \mathbf{q})
\end{pmatrix}
\]

updates for medium parameters
updates for all fields

How to solve?
Quadratic-penalty based full space methods

Initial attempt: block diagonal approximation

\[
\begin{pmatrix}
PP + \lambda^2 HH & 0 \\
0 & \lambda^2 G_m^* G_m
\end{pmatrix}
\begin{pmatrix}
\delta u \\
\delta m
\end{pmatrix}
= -
\begin{pmatrix}
PP(Pu - d) + \lambda^2 HH(Hu - q) \\
\lambda^2 G_m^*(Hu - q)
\end{pmatrix}
\]

Give up some of Newton’s method properties.

Update computation intrinsically parallel per field.

Same as WRI

- if system is solved exactly, and
- fields are initialized by data fit via exact solve data-augmented system
Gradients

First update FWI

First update WRI, $\lambda = 1$
Toy problem

- cross-well setting
- 4 frequencies [6-10] Hz
- 5 simultaneous sources
- 5 receivers
Toy problem

Initial guess:

- $x \ [m]$
  - 0
  - 200
  - 400

- $z \ [m]$
  - 0
  - 200
  - 400

- Velocity $\ [m/s]$
  - 2200
  - 2400
  - 2600
  - 2800
  - 3000
  - 3200
  - 3400

True model:

- $x \ [m]$
  - 0
  - 200
  - 400

- $z \ [m]$
  - 0
  - 200
  - 400

- Velocity $\ [m/s]$
  - 2200
  - 2400
  - 2600
  - 2800
  - 3000
  - 3200
  - 3400
Toy problem

direct solve, full space, $\lambda=1000$

direct solve, reduced space, $\lambda=1000$

direct solution for least-squares problems
Toy problem

iterative solve, full space, $\lambda=1000$

iterative solve, reduced space, $\lambda=1000$

accurate iterative solution for least-squares problems
Toy problem

Iterative solve, full space, $\lambda=1000$

Iterative solve, reduced space, $\lambda=1000$

Inaccurate iterative solution for least-squares problems
Toy problem

- cross-well setting
- 4 frequencies [4-10] Hz
- 20 sources, 9 receivers
Results

Result full-space, 100 iter/subproblem

Result full-space, 75 iter/subproblem

Result full-space, 50 iter/subproblem

Result WRI, 300 iter/subproblem

Result WRI, 200 iter/subproblem

Result WRI, 100 iter/subproblem
Results:
model error for various iteration budgets for sub-problems

Reduced–space quadratic penalty

Full–space quadratic penalty

no descent direction found for inaccurate sub-problem solves
Memory requirements

Keep in memory all fields for all frequencies & sources
Can be distributed over multiple nodes
Quadratic-penalty based full-space avoids storage of multipliers, but may not be enough by itself

Feasible? Need
- parallel computing
- simultaneous sources
- stochastic optimization to reduce the working memory
- small frequency batches
# Full vs Reduced-space

<table>
<thead>
<tr>
<th></th>
<th>Reduced-space FWI</th>
<th>Reduced-space WRI</th>
<th>Full-space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hessian, gradient &amp; function evaluation</strong></td>
<td>solve PDE’s</td>
<td>solve 1 least-squares problem</td>
<td>~free</td>
</tr>
<tr>
<td><strong>Hessian, gradient &amp; function evaluation</strong></td>
<td>inexact</td>
<td>inexact</td>
<td>exact</td>
</tr>
<tr>
<td><strong>Hessian</strong></td>
<td>dense</td>
<td>sparse + dense</td>
<td>sparse</td>
</tr>
<tr>
<td><strong>memory for fields</strong></td>
<td>2 fields per parallel process</td>
<td>1 field per parallel process</td>
<td>all fields in memory (can be distributed over nodes)</td>
</tr>
<tr>
<td><strong>working memory</strong></td>
<td>1 gradient &amp; update direction</td>
<td>1 gradient &amp; update direction</td>
<td>update directions &amp; gradients in memory</td>
</tr>
</tbody>
</table>

~free = sparse matrix-vector products
Conclusions

WRI’s extends the search space
  - less reliant on starting models
  - but requires accurate solves

All-at-once methods remain memory intensive
  - but less reliant on accurate solves
  - still need accurate initialization wavefields by guaranteeing data fits

*Bottom line. Not clear which one will perform better... yet.*
Acknowledgements

Thanks to Chevron and thank you for your attention!

https://www.slim.eos.ubc.ca/
References


