Reverse Time Migration: Checkpointing and Scaling

William W. Symes

TRIP Annual Review, January 2007

Agenda

- RTM as adjoint state method
- Checkpointing
- Griewank's optimal schedule
- Implications for RTM
- Scaling as substitute for inversion - How to make it work
- A practical filtering-scaling algorithm
- Example - Marmousmooth

Discrete Time Evolution

Dynamical operator $\mathbf{H}^n$ depends on *control* $\mathbf{c}$, advances *state* $\mathbf{u}^n$ one time step.

$$\mathbf{u}^{n+1} = \mathbf{H}^n[\mathbf{c}, \mathbf{u}^n],\ n = 0, 1, ..., N-1$$

Main example for this talk: constant density acoustic wave equation approximated by centered differences. Acoustic potential $u^n_{ijk} \simeq u(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$, with velocity field $v_{ijk} \simeq v(i\Delta x, j\Delta y, k\Delta z)$, source $f^n_{ijk} \simeq$ ..., L = finite difference or element approximation to Laplacian:

$$\mathbf{u}^n = \left(\begin{array}{c} u^n \\ u^{n-1} \end{array} \right),\ \mathbf{c} = (v),\ \mathbf{H}^n[\mathbf{c}, \mathbf{u}^n] = \left(\begin{array}{c} 2u^n - u^{n-1} + \Delta t^2 v^2 L u^n + \Delta t^2 f^n \\ u^n \end{array} \right)$$

Same ideas/formalism applies to other schemes (staggered grid, FEM,...), models (elasticity, viscoelasticity,...), seismic apps (WEMVA - Shen et al. SEG 03, Biondi-Sava 04, Soubaras 06), physics (EM, heat flow, weather system, ocean currents,...)

Objective or Cost Functions

$\mathbf{u}[\mathbf{c}] = (\mathbf{u}^0, \mathbf{u}^1, ..., \mathbf{u}^N)^T \in U^N$ = state time series - implicitly function of $\mathbf{c}$.

$\mathbf{S} : U^N \rightarrow E$ = *sampling operator*.

For this talk: E = seismic traces (pressure $\partial u/\partial t$ sampled in space/time).

$\mathbf{G} : E \rightarrow \mathbf{R}$ = *"goodness" function*, defines *cost* function $J : C \rightarrow \mathbf{R}$ via $J[\mathbf{c}] = \mathbf{G}[\mathbf{S}[\mathbf{u}[\mathbf{c}]]]$.

Classic example leading to RTM: $\mathbf{G}$ = mean square error function, i.e. $\mathbf{d} \in E$ = data, $\mathbf{G}[\mathbf{s}] = \frac{1}{2}\|\mathbf{d} - \mathbf{s}\|^2$.

Adjoint State Method

For computing the gradient of J:

- compute $\mathbf{u}[\mathbf{c}] = (\mathbf{u}^0, ..., \mathbf{u}^N)^T$ ("forward sweep"), initialize *gradient accumulator* $\mathbf{g}^N \in C$ and *adjoint state* $\mathbf{w}^{N+1} \in U$ to zero.
- For $n = N-1, ...0$ ("backwards sweep"),

$$\begin{aligned} \mathbf{w}^{n+1} &= D_u\mathbf{H}^{n+1}[\mathbf{c}, \mathbf{u}^{n+1}]^T\mathbf{w}^{n+2} + [\mathbf{S}^T(\nabla\mathbf{G})[\mathbf{S}[\mathbf{u}[\mathbf{c}]]]]^{n+1} \\ \mathbf{g}^n &= \mathbf{g}^{n+1} + D_c\mathbf{H}^n[\mathbf{c}, \mathbf{u}^n]^T\mathbf{w}^{n+1} \end{aligned}$$

- $\nabla J[\mathbf{c}] = \mathbf{g}^0$.

For acoustics, $\mathbf{w}^n = (w^n, w^{n+1})^T$, and $D_c\mathbf{H}^n[\mathbf{c}, \mathbf{u}^n]^T\mathbf{w}^{n+1} = 2\Delta t^2 v(Lu^n)w^{n+1}$ - cross correlation of incident (u), backpropagated (w) fields, $\nabla J[\mathbf{c}] = \mathbf{g}^0$ = image.

Computational Complexity

Observation: $\mathbf{u}$ evolves *forward* in step index, $\mathbf{w}$ *backward* in step index, but they are needed at indices $n, n+1$ respectively, $n = N-1, ...0$.

Strategies for simultaneous access to $\mathbf{u}^n, \mathbf{w}^{n+1}$ – in all cases $\mathbf{w}^{n+1}$ evolved backwards from $n = N$ to $n = 0$.

1. For each n, evolve $\mathbf{u}^n$ from $n = 0$.
2. Compute $\mathbf{u}^0, ..., \mathbf{u}^N$, store all; For each n retrieve $\mathbf{u}^n$.
3. Compute $\mathbf{u}^0, ...\mathbf{u}^N$, store every kth state, $k > 1$; for each n, interpolate n state from closest stored states. Used in some commercial 2D RTM implementations.
4. Compute $\mathbf{u}^0, ...\mathbf{u}^N$, evolve $\mathbf{u}^n$ backwards in time from $n = N$. Possible for acoustic RTM, if enough boundary data stored to make up for ABC. Not available for attenuative modeling, reasonable Q.

Computational Complexity

Cost: units of simulation steps (flops) to compute $\mathbf{u}$, number of state vectors stored:

1. working storage (1 state vector), $N^2/2$ steps - prohibitive;
2. N steps, N state vectors;
3. also N steps, N/k state vectors, but loss of accuracy due to use of interpolation rather than evolution;
4. $2N$ steps, 1 state vector. For acoustic RTM with ABC - add'l storage equivalent to 10's of state vectors.

3D RTM: $N \simeq 10000$, state vector $\simeq 10^9$ W $\Rightarrow$ (1) strategy 1 $\sim O(10^{38})$ flops, (2) strategy 2 $\sim O(20-40)$ TB, strategy 3 $\sim O(2-4)$ TB w/ $k = 10$.

There's a better way...

Checkpointing

Alternative to strategies 1-4. Requires allocation of

- N_B buffers, each storing one state vector;
- $N_C >> N_B$ checkpoints = integers between 0 and N.

Forward sweep (n=0,...,N): solve forward evolution problem to compute $\mathbf{u}^0, \ldots, \mathbf{u}^N$; store N_B checkpoints in the buffers, including the first (always n=0) and last.

Backwards sweep (n=N-1,...,0): begin by using strategy 1, *starting at the last checkpoint*. When the n = last checkpoint, re-use its buffer to store another checkpoint. computing its state by application of strategy 1 starting from the previous stored checkpoint. Continue using strategy 1, starting from next-to-last checkpoint [this must be the replacement for the last checkpoint, unless it was previously stored]. Continue. At end of algorithm, buffers store some number of states starting with $n = 0$; finish using strategy 2.

Checkpointing

Example with $N = 15, N_B = 3, N_C = 6$

Meaning of colums:

- bufk records checkpoint stored in buffer k;
- *recomp* records the previously computed steps which are *recomputed* in each step of the backwards sweep, or *dash* if no recomputation necessary in step;
- *bold faced* checkpoints used as Cauchy data for strategy 1;
- *italic*: n for which $\mathbf{u}^n$ combined with $\mathbf{w}^{n+1}$ in evaluation of gradient update.

During forward sweep checkpoints 0, 6, 11 recorded in buffers 1, 2, and 3.

step	buf1	buf2	buf3	recomp
14	0	6	**11**	12,13,*14*
13	0	6	**11**	12, *13*
12	0	6	**11**	*12*
11	0	**6**	*11*	7, 8
10	0	6	**8**	9, *10*
9	0	6	**8**	*9*
8	0	6	*8*	-
7	0	**6**	8	*7*
6	0	*6*	8	-
5	**0**	1	3	1, 2, 3, 4, *5*
4	0	1	**3**	*4*
3	0	1	*3*	-
2	0	**1**	3	*2*
1	0	*1*	3	-
0	*0*	1	3	-

Griewank's Optimal Checkpoint Schedule

Big question: how do you choose checkpoints to

- minimize the amount of recomputation for given storage allocation (N_B), or
- minimize the amount of storage required for a given level of recomputation.

Solution by Griewank, *Opt. Meth. and Software*, 1992, published as Alg. 799, Griewank and Walther, *ACM TOMS* 2000, in terms of *recomputation ratio* = total number of forward steps required to compute adjoint / N.

$$N = 10000$$

buffers	3	5	10	15	20	25	30	35	40	60
ratio	27.9	11.3	5.8	4.5	3.8	3.6	3.4	3.1	2.9	2.8

Implications for 3D RTM

$N = 10000$, buffers for 36 state vectors $\Rightarrow$

- total cost of adjoint $\simeq$ 3 times forward simulation + 1.5 times for adjoint step $(\mathbf{w}^{n+1} \mapsto \mathbf{w}^n) \simeq$ 4.5 times sim cost.
- total storage required $\simeq 150$ GB (compare 2 TB for strat. 3, 20 TB for strat 2, both at 2.5 times sim cost).
- with optimal checkpointing *in-core* 3D RTM feasible now on *subclusters* (eg. 8GB - 1 Gflop $\Rightarrow$ several shots/day on 20 nodes)
- alternative - store checkpoints to disk - i/o cost reduced by 1-2 ord. of magnitude.
- store less reference state $\Rightarrow$ either do less i/o or have more core available for working fields $\Rightarrow$ fewer nodes needed, less message passing.
- multicore/stream FPUs (Cell, GPU, FPGA,...) $\Rightarrow$ advantage: checkpointing

TRIP 2D RTM

- available to sponsors since 5/06.
- Features: (2,4) FD scheme, PML ABC's, minimal optimization, models in SEP77, data in SU/SEGY, RVL/TSOpt C++/MPI framework, incorporates Griewank-Walther checkpoint scheduling, F77 loops processed with TAMC (AD).
- no message passing, no disk i/o within time loop.
- Expl: derived from Marmousi, 240 shots, 3 s, 800x2500 (x, z) grid, ~ 8000 time steps, parallelized over shot. On 120 cores of Rice Cray XD-1, gcc4: Modeling = 20 min, RTM = 90 min.

Migration vs. Inversion

RTM produces *gradient* of least squares cost function: if data is $\mathbf{d}$, and $\mathbf{F}$ is *forward map*

$$\mathbf{F}[\mathbf{c}] = \mathbf{S}[\mathbf{u}[\mathbf{c}]]$$

then

$$\nabla J[\mathbf{c}] = D\mathbf{F}[c]^T(d - \mathbf{F}[c])$$

RTM output ∇J is *image* under certain circumstances: if *Born approximation*

$$\mathbf{d} \simeq \mathbf{F}[\mathbf{c}_0] + D\mathbf{F}[\mathbf{c}_0]\delta\mathbf{c}$$

is accurate, and $\mathbf{c}_0$ is known and "nonreflecting", then $\nabla J[\mathbf{c}_0]$ is an image.

BUT it's not an inversion, i.e. generally $\nabla J[\mathbf{c}_0] \neq \delta\mathbf{c} \simeq (D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0])^{-1}\nabla J[\mathbf{c}_0]$.

Example: Marmousmooth

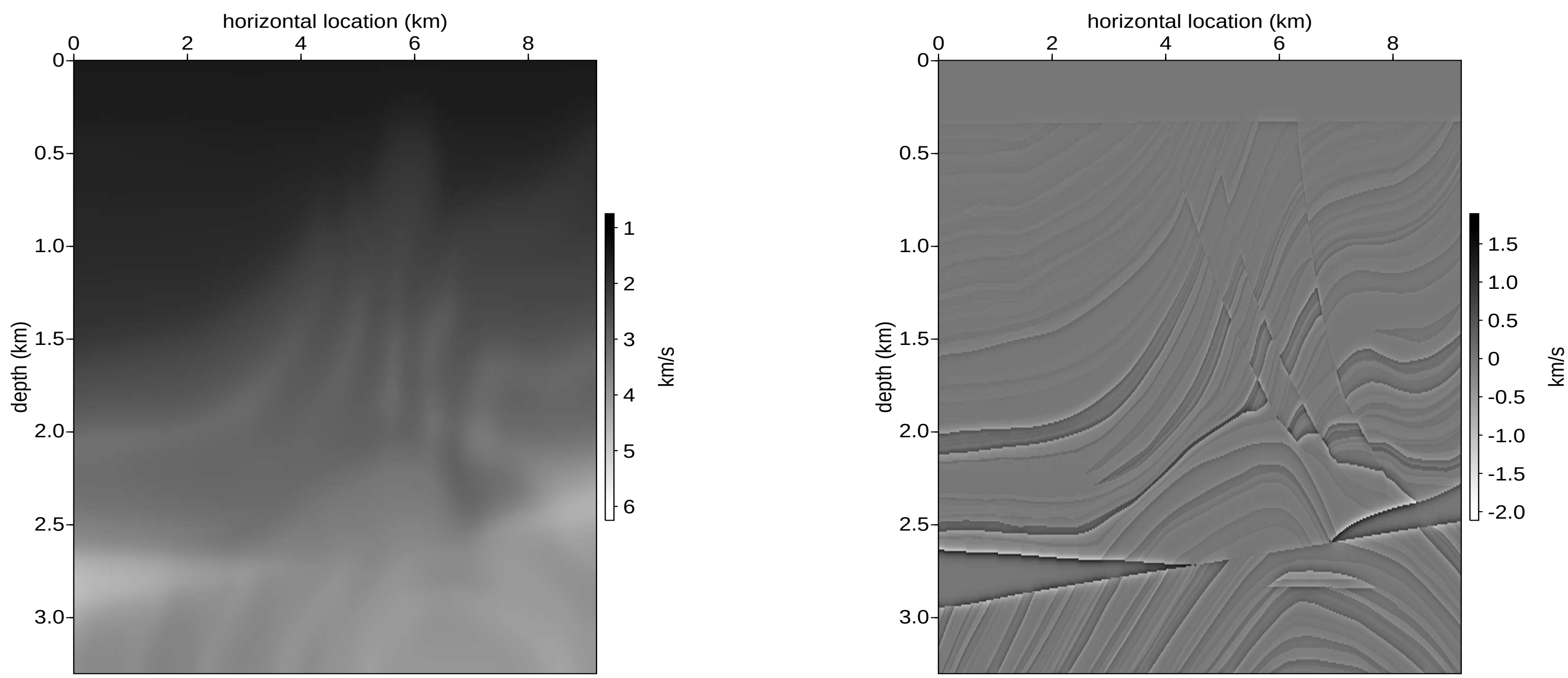

Figure 1. Left: Marmousi velocity model smoothed with tapered 160 m radius moving average. Right: Velocity perturbation, difference of original Marmousi model and 40 m smoothing.

Example: Marmousmooth

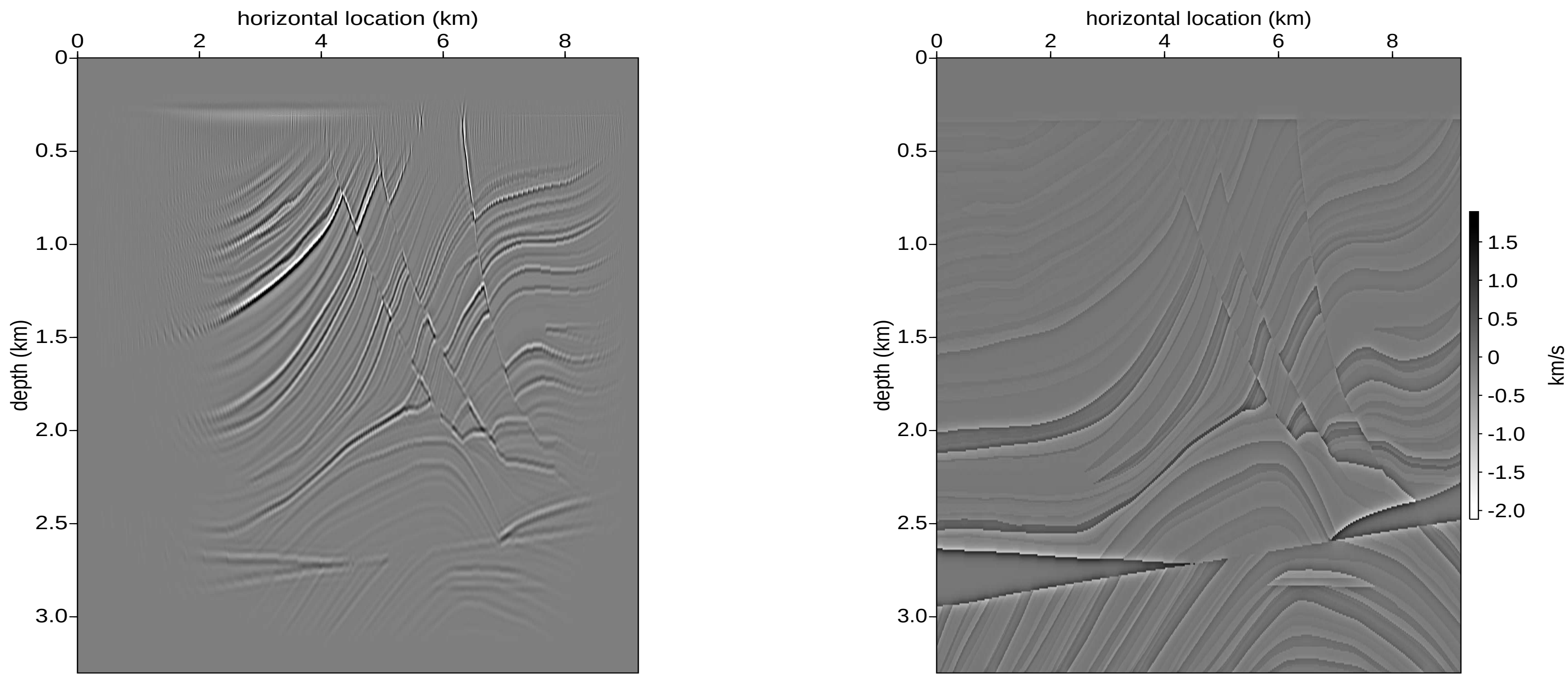

Figure 2. Left: RTM of Born data created from model of Figure 1, source = 5-13-40-55 bandpass. Right: Velocity perturbation, displayed for comparison. Note discrepancy between shallow and deep amplitudes in image vs. model structure.

Scaling as Approximate Inversion

Claerbout-Nichols (SEP 82, 94): necessarily $(D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0])^{-1}\nabla J[\mathbf{c}_0] \simeq s\nabla J[\mathbf{c}_0]$ for spatially varying $s(\mathbf{x})$. Estimate s by solving a related least-squares problem, say

$$\nabla J[\mathbf{c}_0] \simeq s(D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0])\nabla J[\mathbf{c}_0]$$

Both sides of above are computable (one additional Born modeling ("demigration") and migration).

Rickett (*Geophys.* 03) applied this idea to shot-profile migration.

However it won't work... unless $D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0]$ is approximately diagonal, i.e. multiplication by a function, which it is not! [Easy counterexamples!]

Guitton (*Geophys.* 04) replaces scale factor s by *spatially varying filter* - better results, but structure of this "filter" unclear - how many degrees of freedom?

How to make it work (1)

Structure Theorem for Born Modeling: (Beylkin, 85; Rakesh, 86; Nolan & Symes 97; Smit, tenKroode, & Verdel 98; Stolk 00) Generally,

$$D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0]\chi(\mathbf{x})e^{i\omega\psi(\mathbf{x})} = \sigma(\mathbf{x},\omega\nabla\psi(\mathbf{x}))\chi(\mathbf{x})e^{i\omega\psi(\mathbf{x})} + O(|\omega|^{m-\beta})$$

where:

- χ smooth, vanishing outside ball of radius > 0, $\nabla\psi(\mathbf{x}) \neq 0$ if $\chi(\mathbf{x}) \neq 0$;
- $\sigma(\mathbf{x},\mathbf{k}) \geq 0$ is homogeneous of degree m in $\mathbf{k}$;
- $\beta > 0$.

Operators with this property (acting as a multiplier on localized monochromatic pulses) are *pseudodifferential* (“ΨDO”). *Order* is $m = d - 1$ in space dimension d, σ is *principal symbol*.

How to make it work (2)

Key facts about operators:

Differential operators are ΨDOs, but not all ΨDOs are differential - for example, arbitrary real powers of Laplace op are ΨDO.

ΨDOs form an *algebra*: sums and products are ΨDOs. Product commutes modulo lower order ops: principal symbol of product is product of principal symbols.

$\Rightarrow (-\nabla^2)^{-\frac{m}{2}} D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0]$ is an operator of order 0.

Operators of order zero act as *frequency independent* multipliers on monochromatic pulses:

$$(-\nabla^2)^{-\frac{m}{2}} D\mathbf{F}[\mathbf{c}_0]^T D\mathbf{F}[\mathbf{c}_0]\chi(\mathbf{x})e^{i\omega\psi(\mathbf{x})} = \bar{\sigma}(\mathbf{x})\chi(\mathbf{x})e^{i\omega\psi(\mathbf{x})} + O(|\omega|^{-\beta})$$

where $\bar{\sigma}(\mathbf{x}) = \|\nabla\psi(\mathbf{x})\|^{-m}\sigma(\mathbf{x}, \nabla\psi(\mathbf{x}))$.

How to make it work (3)

Key fact about images:

Seismic images (migration outputs), and presumably the models to which they correspond, tend to have well-defined dip in most places, i.e. to be local Fourier sums of monochromatic pulses. So: in most places,

- migrating data, then filtering the migration output by $(-\nabla^2)^{-\frac{m}{2}}$ gives (approx.) δv multiplied by $\bar{\sigma}$;
- remodeling the data (applying $D\mathbf{F}[\mathbf{c}_0]$ to the migration output), then remigrating this remodeled data, then filtering, gives migrated image multiplied by same $\bar{\sigma}$.

Use second relation to estimate $\bar{\sigma}$, then first to estimate δv (divide by $\bar{\sigma}$): turns a migration into an inversion. Same idea as Claerbout-Nichols 94 and Rickett 03, but with additional filtering step; similar to Guitton 04, but operator structure fully specified.

A Practical Filtering-Scaling Algorithm

1. perform prestack migration $\mathbf{d} \mapsto D\mathbf{F}[\mathbf{c}_0]^T\mathbf{d} \equiv \mathbf{c}_{\rm mig}$;
2. resimulate the data: $\mathbf{c}_{\rm mig} \mapsto D\mathbf{F}[\mathbf{c}_0]\mathbf{c}_{\rm mig} \equiv \mathbf{d}_{\rm resim}$;
3. remigrate the resimulated data: $\mathbf{d}_{\rm resim} \mapsto D\mathbf{F}[\mathbf{c}_0]^T\mathbf{d}_{\rm resim} \equiv \mathbf{c}_{\rm remig}$;
4. Apply the Laplace filter: $\mathbf{c}_{\rm remig} \mapsto L^{-\frac{m}{2}}\mathbf{c}_{\rm remig} \equiv \mathbf{c}_{\rm filt}$ (here $-L$ is an approximation to the Laplace op);
5. Find a nonnegative scale factor $\mathbf{W}^2$ for which $\mathbf{W}^2\mathbf{c}_{\rm filt} \simeq \mathbf{c}_{\rm mig}$ ($\mathbf{W}^2 =$ pseudoinverse op to multiplication by $\bar{\sigma}$);
6. Compute the approximate inverse $\mathbf{c}_{\rm est} = \mathbf{W}^2 L^{-\frac{m}{2}}\mathbf{c}_{\rm mig}$.

[For comparison: Claerbout-Nichols-Rickett alg is *same*, except leave out the filtering steps.]

Example: Marmousmooth again

Details of implementation:

1. Used TRIP 2D RTM package, which includes Born modeling.

2. Laplace filter $(-\nabla)^{-\frac{1}{2}}$ implemented via 2D FFT.

3. Determine $\mathbf{W}^2$ = multiplication by $\bar{\sigma}^\dagger$ by solving nonlinear least squares problem for $\tau = \log(\bar{\sigma}^\dagger)$ - simple device to ensure that computed $\mathbf{W}^2$ is positive definite. Used RVL/Alg implementation of LBFGS.

4. To avoid migration aperture edge artifacts, focussed on central region of model via tapered spatial mute ("cutoff function").

Velocity Pert. vs. Scaling-Filtering Inversion

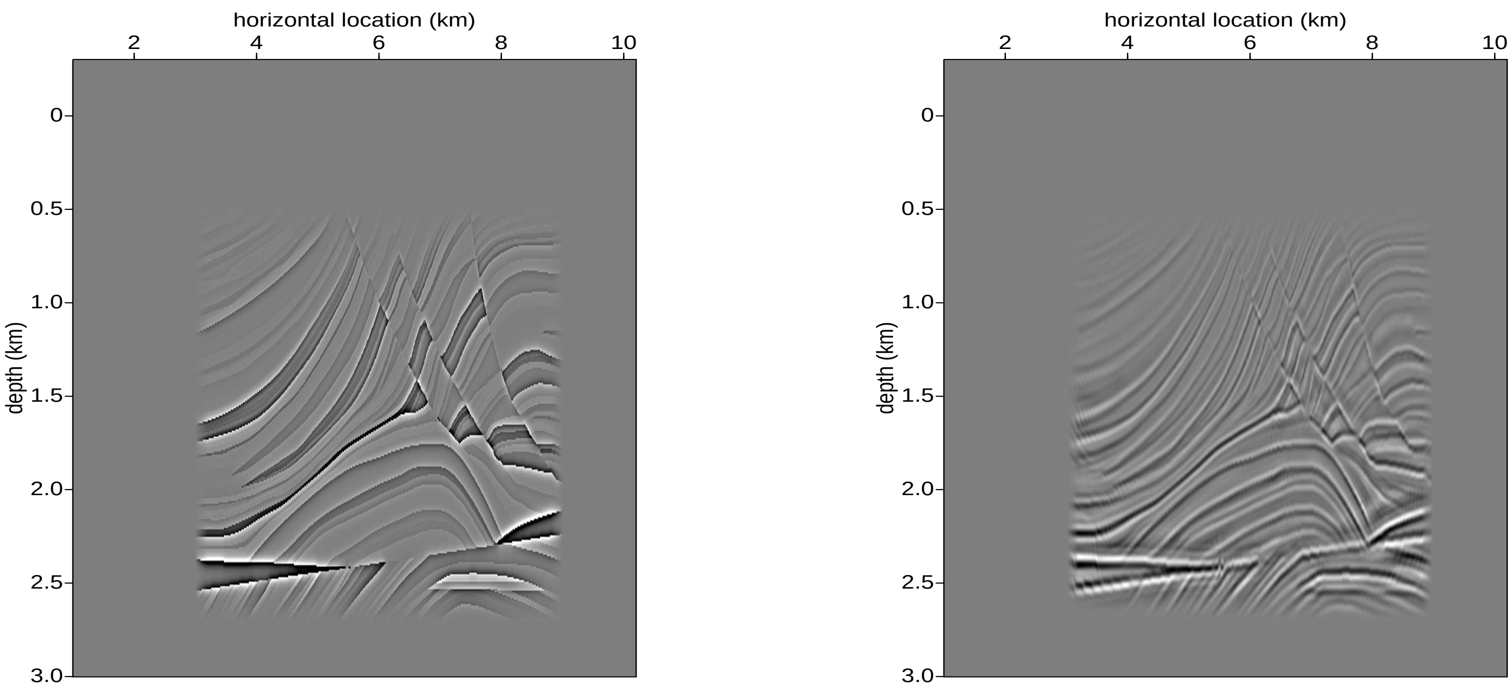

Figure 3. Left: Velocity perturbation, difference of original Marmousi model and 40 m smoothing. Right: Approximate inversion from Scaling-Filtering Algorithm.

Velocity Pert. vs. Scaling Only

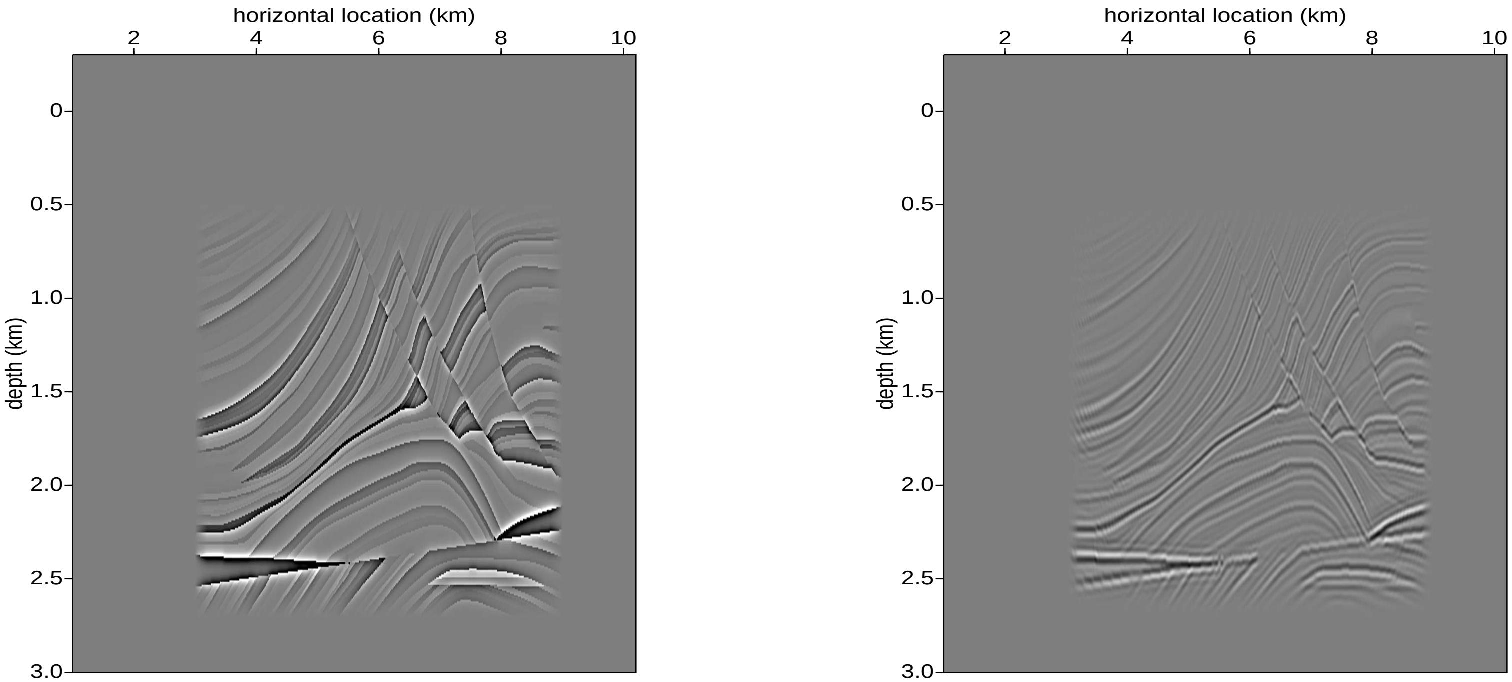

Figure 4. Left: Velocity perturbation, difference of original Marmousi model and 40 m smoothing. Right: Approximate inversion from Scaling-Filtering Algorithm.

Resimulations: Exact vs. Scaling-Filtering

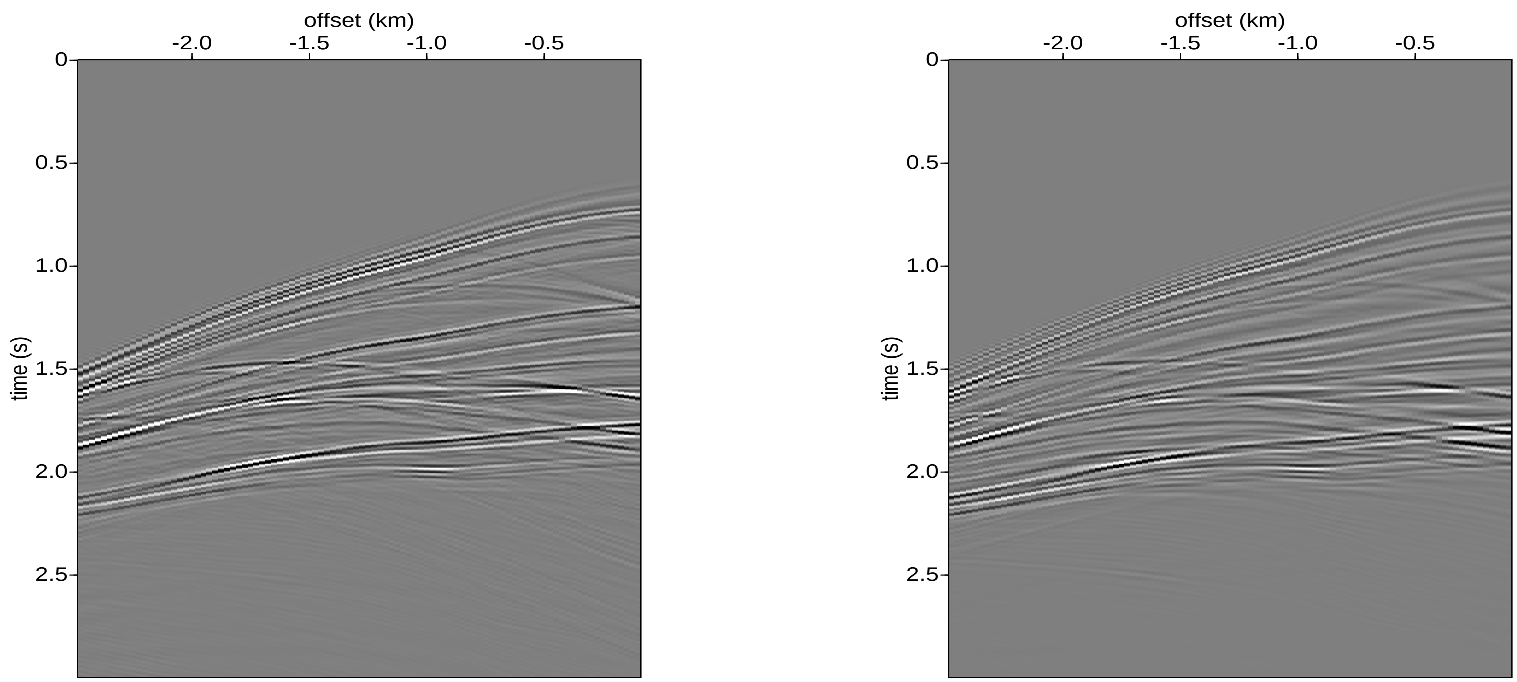

Figure 5. Shot at sx=7500m. Left: Born simulation with exact model, truncated by spatial mute. Right: Born resimulation using scaling-filtering approximate inversion.

Resimulations: Exact vs. Scaling Only

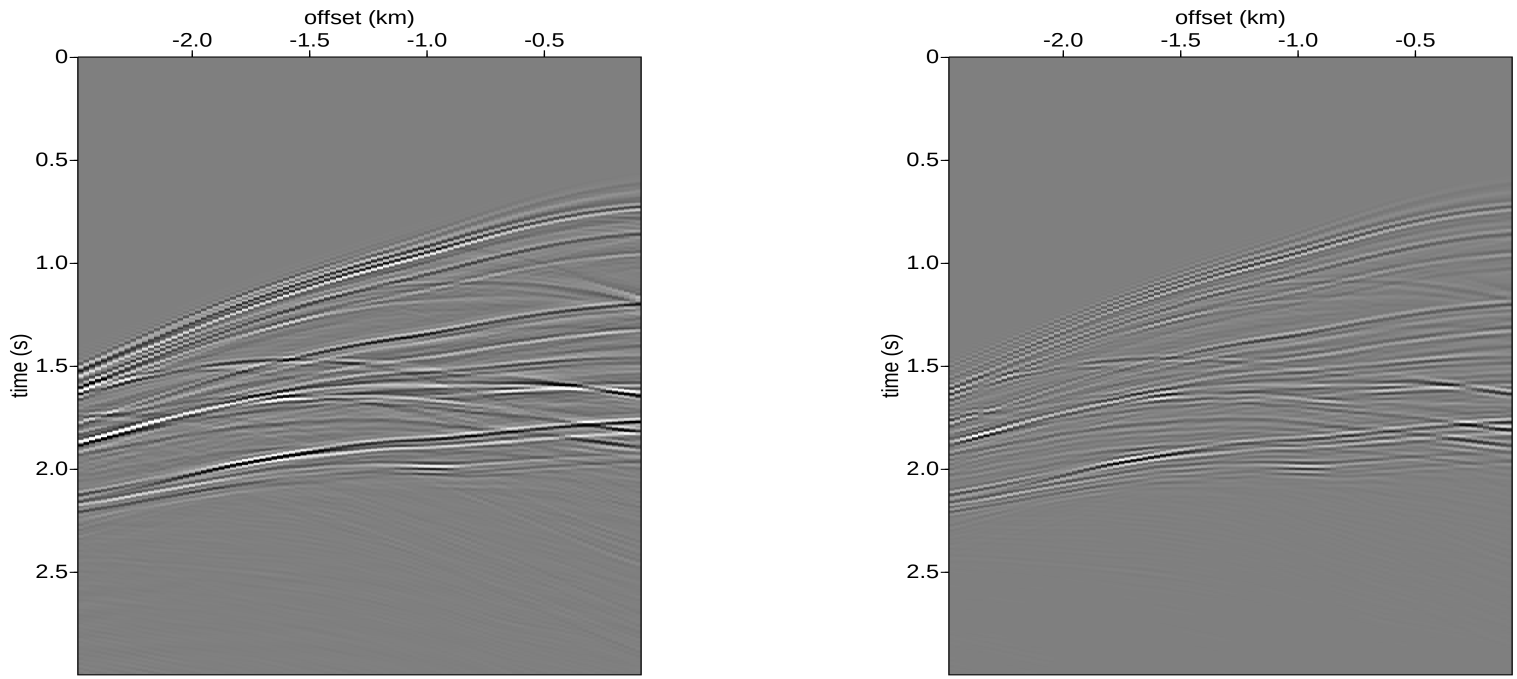

Figure 6. Left: Born simulation with exact model, truncated by spatial mute. Right: Born resimulation using scaling-filtering approximate inversion.

The Smoking Gun: Spectral Comparison

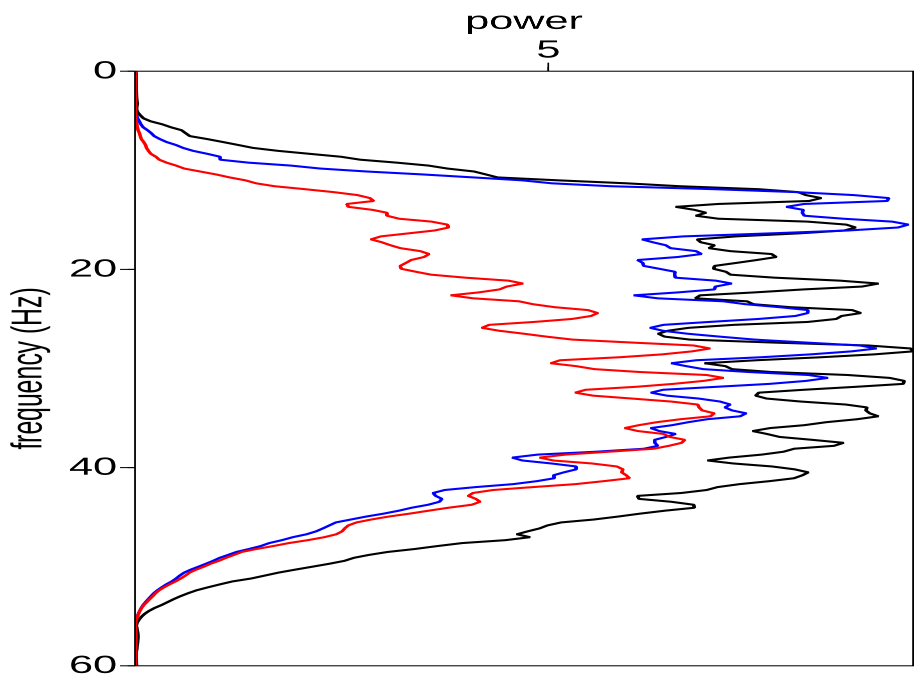

Figure 7. Spectra of simulations for sx=7500m, stacked. Black = true model (truncated by spatial mute), Blue = Scaling-Filtering approx. inversion, Red = Scaling-only approx. inversion. Note missing linear-in-frequency trend in scaling-only result - equivalent to missing division by $|k|$.

Conclusions

- Griewank's optimal checkpointing algorithm dramatically reduces storage required for RTM (by over an order of magnitude) - enabling technology .
- 3D RTM should be possible in-core on modest clusters. 2D RTM requires no intraloop i/o. Checkpointing advantage will increase if flops beat memory (Cell, GPUs,...).
- Approximation of Born inversion by RTM plus scaling requires additional filtering step - cost is extra modeling/migration loop (cost of scale factor estimation is insignificant).
- Scaling-filtering approximate inversion promising as preconditioner for iterative Born inversion using Krylov-type iteration (CG and relatives).
- Theory, practice both require nonreflecting background velocities . Coherent approach to general imaging problems (eg. salt boundary location) appears to require nonlinear inversion.