A multigrid approach for time domain FWI

CREWES Sponsor Meeting

Daniel Trad

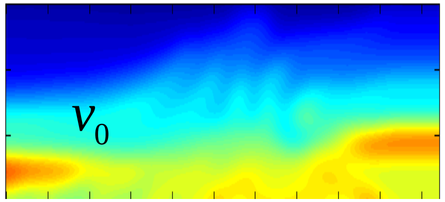
December 4, 2020

NSERC CRSNG
UNIVERSITY OF CALGARY
FACULTY OF SCIENCE
Department of Geoscience

Outline

- **Full waveform inversion basics**
- **Frequency domain vs Time domain**
- **Multigrid approach in the Time domain**
- **Parallel Implementation**
- **CUDA implementations**

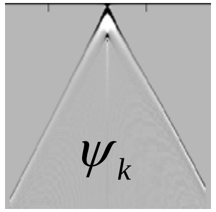
Full Waveform Inversion Basics (Inversion)



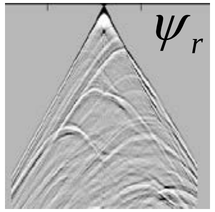
a) Initial velocity v_0 model

$k = 1$

1) Forward model through v_{k-1} to predict data ψ_k



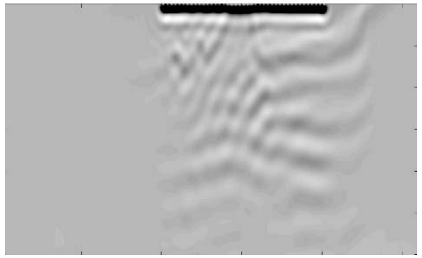
b) Raw data ψ_r



−

$\delta\psi_k = \psi_r - \psi_k$

2) Ungained RTM of “data residual” with v_{k-1} and stack

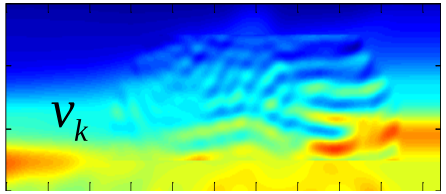


$G(\vec{x}) = \mathrm{RTM}(\delta\psi_k)$

3) Scale gradient (line search) and deduce velocity perturbation δv_k

$\delta v_k = \lambda G(\vec{x})$

4) Update velocity model $v_k = v_{k-1} + \delta v_k$



$k = k + 1$

$$J = \| d_{predicted} - d_{acquired} \|^2$$

$$\mathbf{d}_{\mathbf{predicted}} = \mathbf{L}(\mathbf{v}_{\mathbf{iter}})$$

$$\mathbf{v}_{\mathbf{iter}} = \mathbf{v}_{\mathbf{iter}-1} + \alpha \boldsymbol{\Delta} \mathbf{v}$$

$$\Delta v = RTM(Residuals)$$

Residuals depend on:

1) Velocity errors
2) Modeling errors
3) Processing errors

INVERSE CRIME:

Predictions created with the same algorithm as the data.

consequence:

residuals depend only on velocity.

Time vs Frequency domain implementations

COARSE GRAINED PARALLELIZATION

Cost for n x n x n model:

- model size ~n^3
- number of time steps ~n
- number of sources ~N (~n^2)
- Roughly CPU ~n^4N (~n^6)

Minimize cost by:

- maximize grid spacing
- maximize time step

ISSUE:

Doubling frequency->x 2^3 to 2^4

Parallelization for simultaneous shots

FINE GRAINED PARALLELIZATION

$$\mathbf{A}(\omega, \mathbf{v})\mathbf{p}(\omega, \mathbf{x}) = \mathbf{wavelet}(\omega)$$

Ap=w

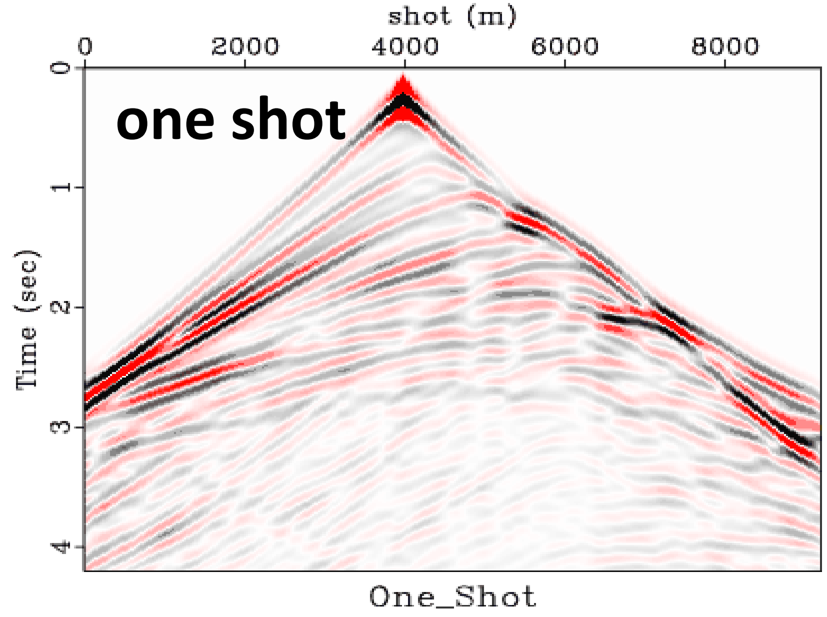
In 3D:

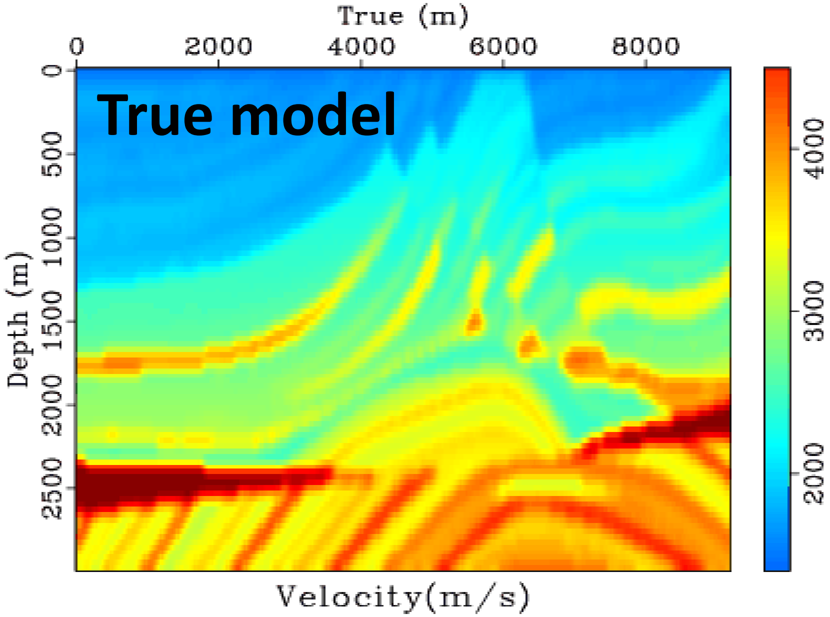
- $\mathbf{A}$ is $n^3 \times n^3$ matrix
- $\mathbf{p}$ & $\mathbf{w}$ are n^3 column vectors
- $\mathbf{A}$ depends on frequency and the model
- $\mathbf{A}$ & $\mathbf{w}$ are known, solve for $\mathbf{p}$
- one equation per frequency
- many shots simultaneously.

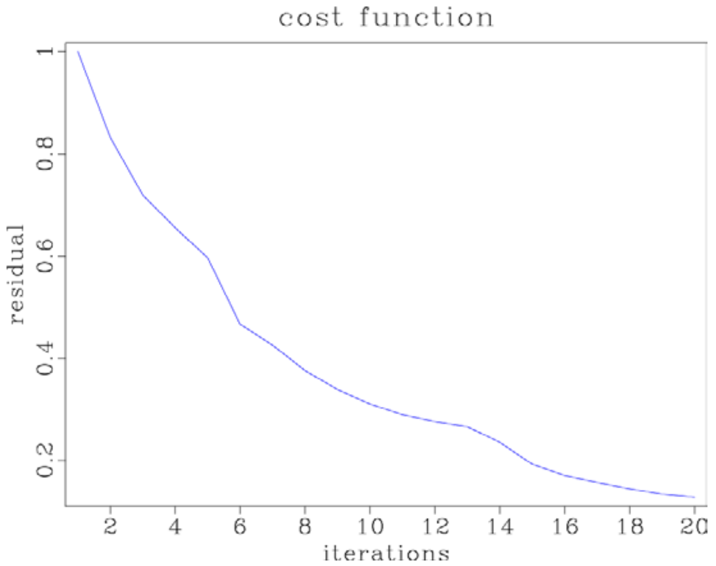
ISSUE: 3D becomes too expensive

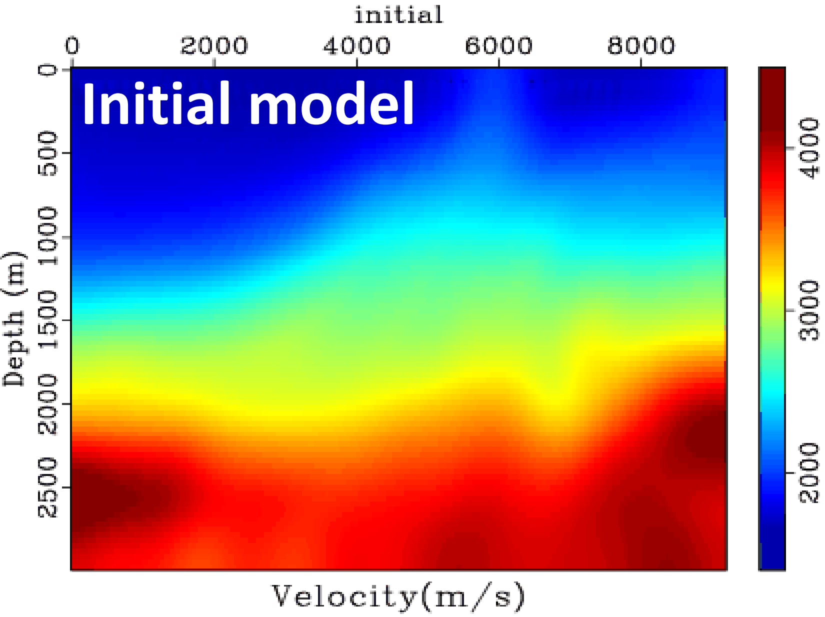
Parallelization for LU decomposition

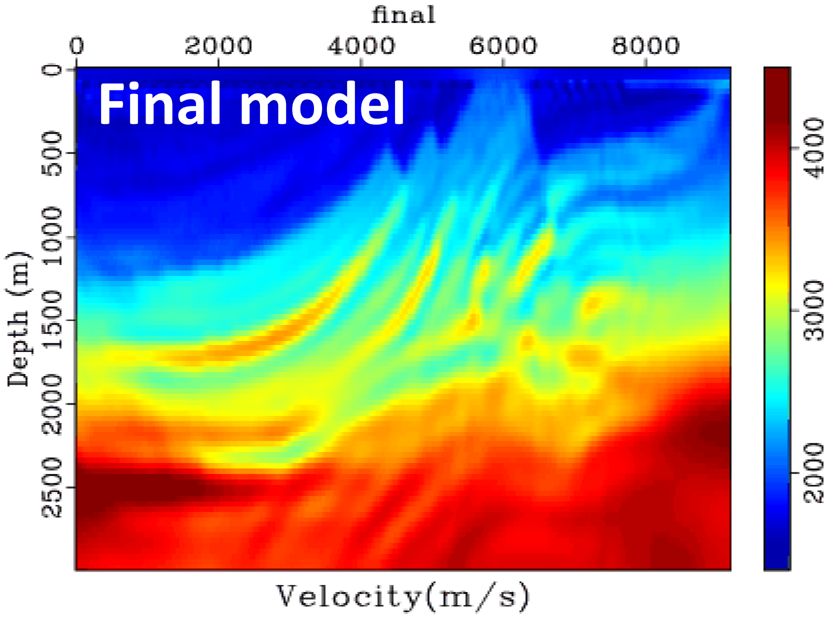
Typical time domain run for Marmousi model





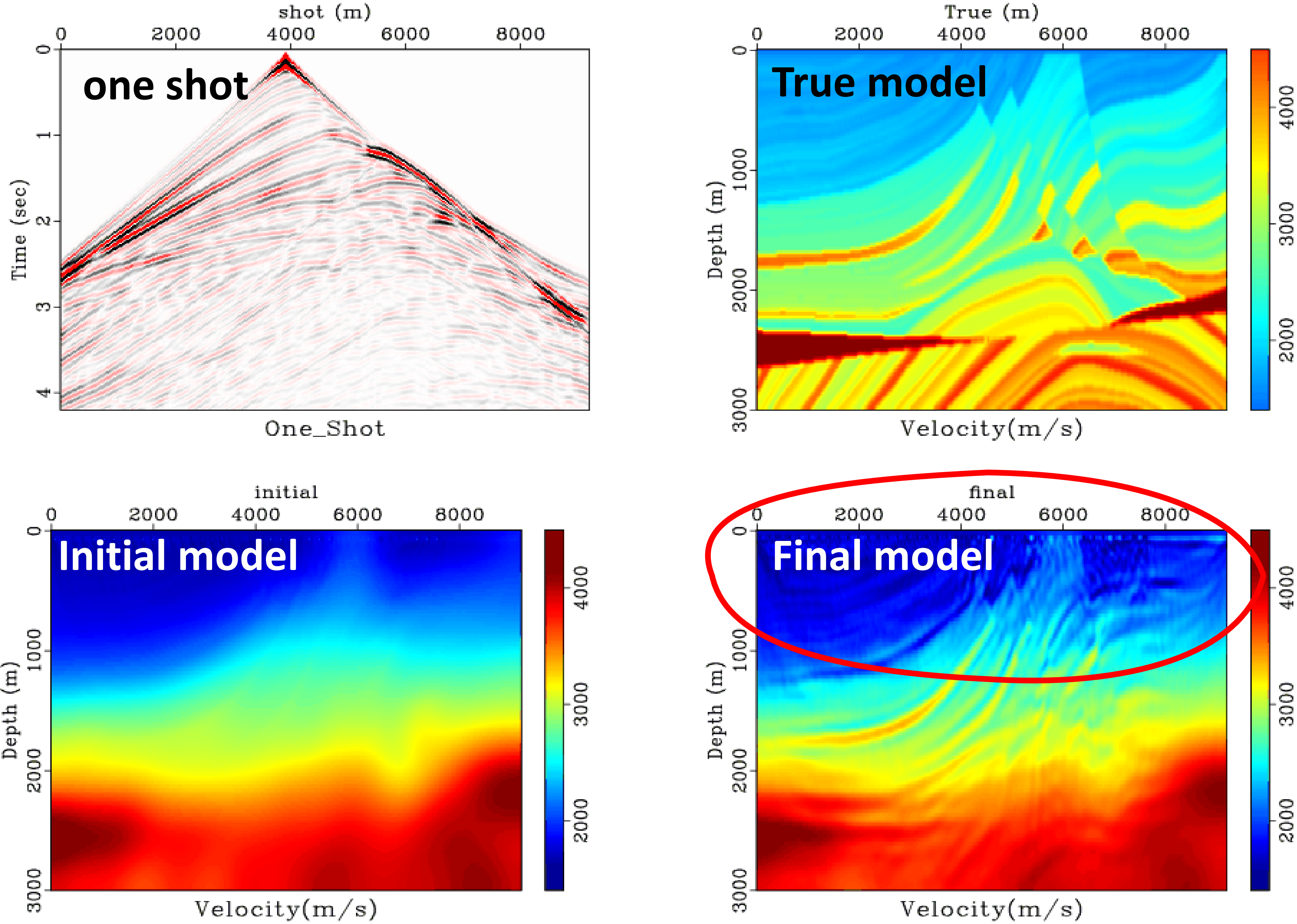






40 shots,
20 iterations
0-8 Hz, 32 x 32 m
5 minutes in 10 nodes

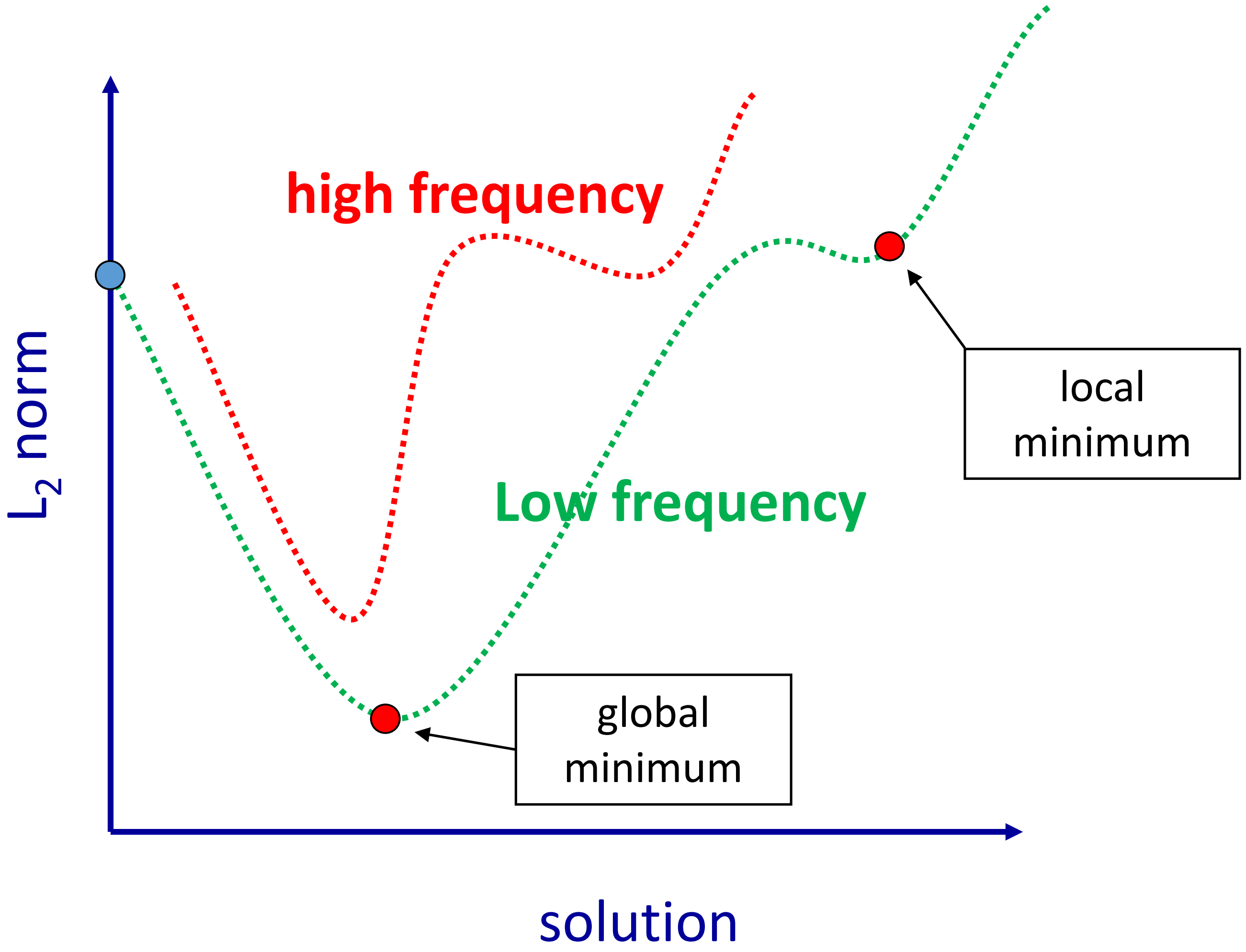
Attempt to double frequency range (0-16Hz)



40 shots,
20 iterations
0-16 Hz, 16 x 16 m

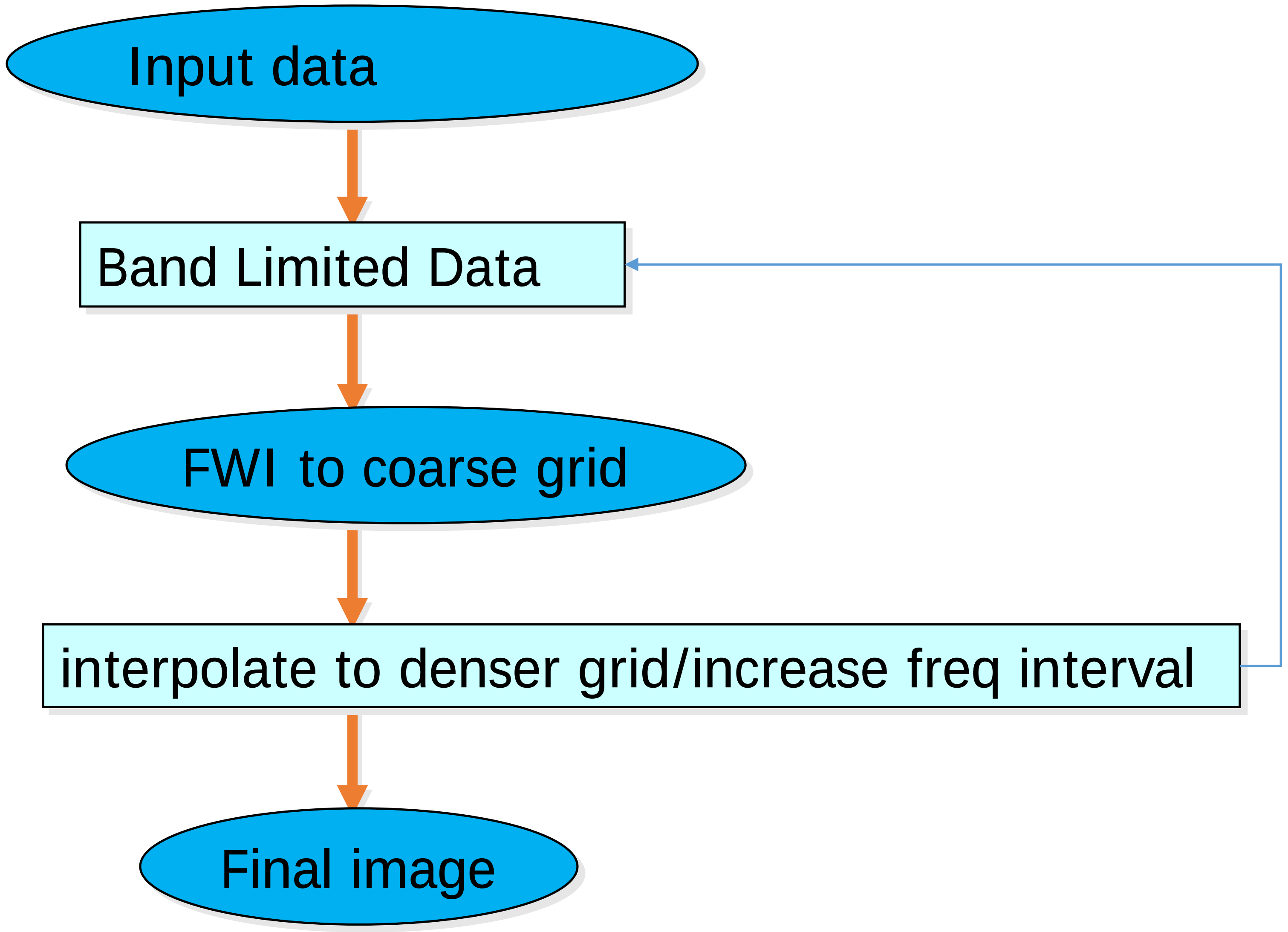
The cycle skipping problem

cost functions for different frequencies

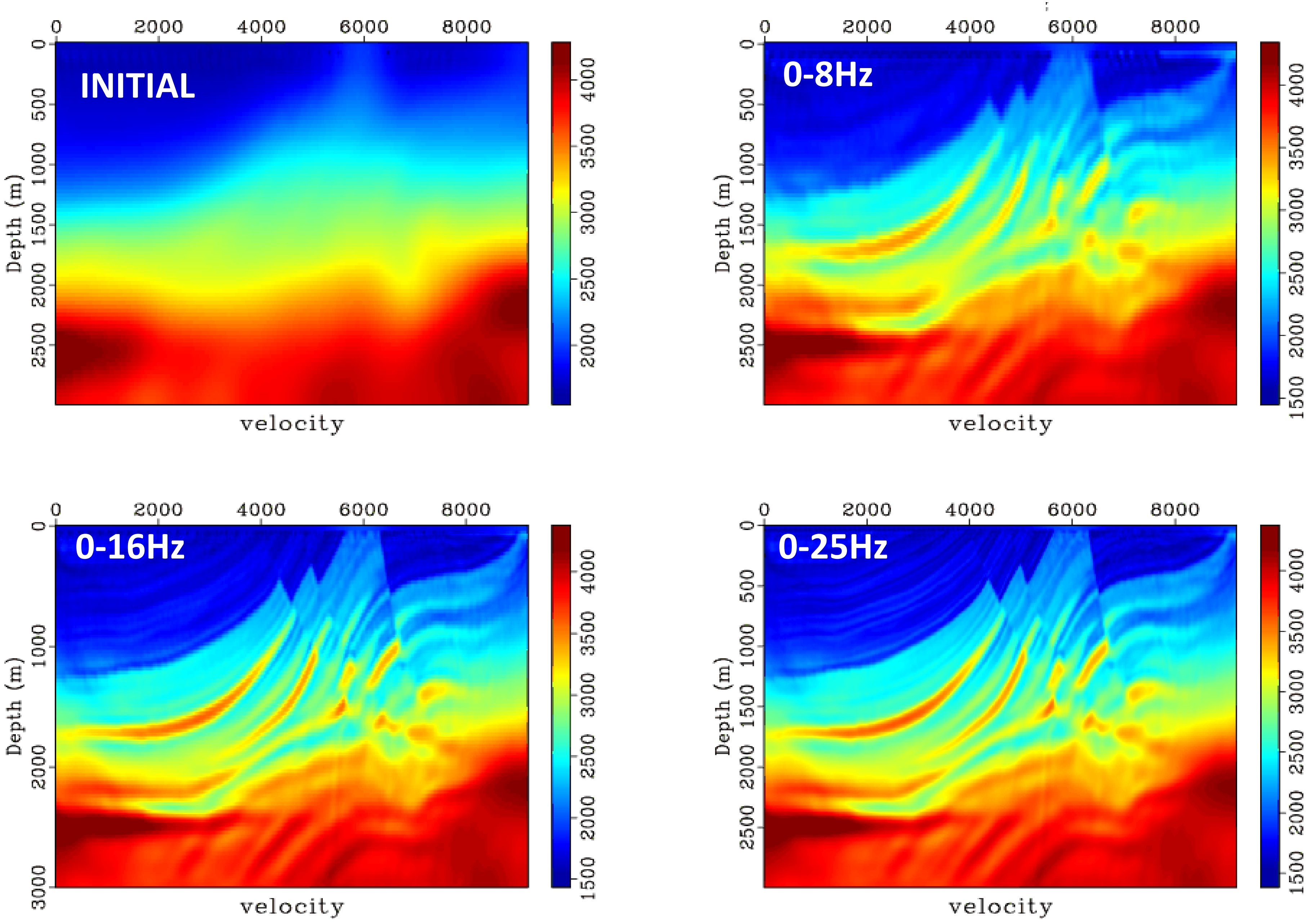


High frequencies require better initial model

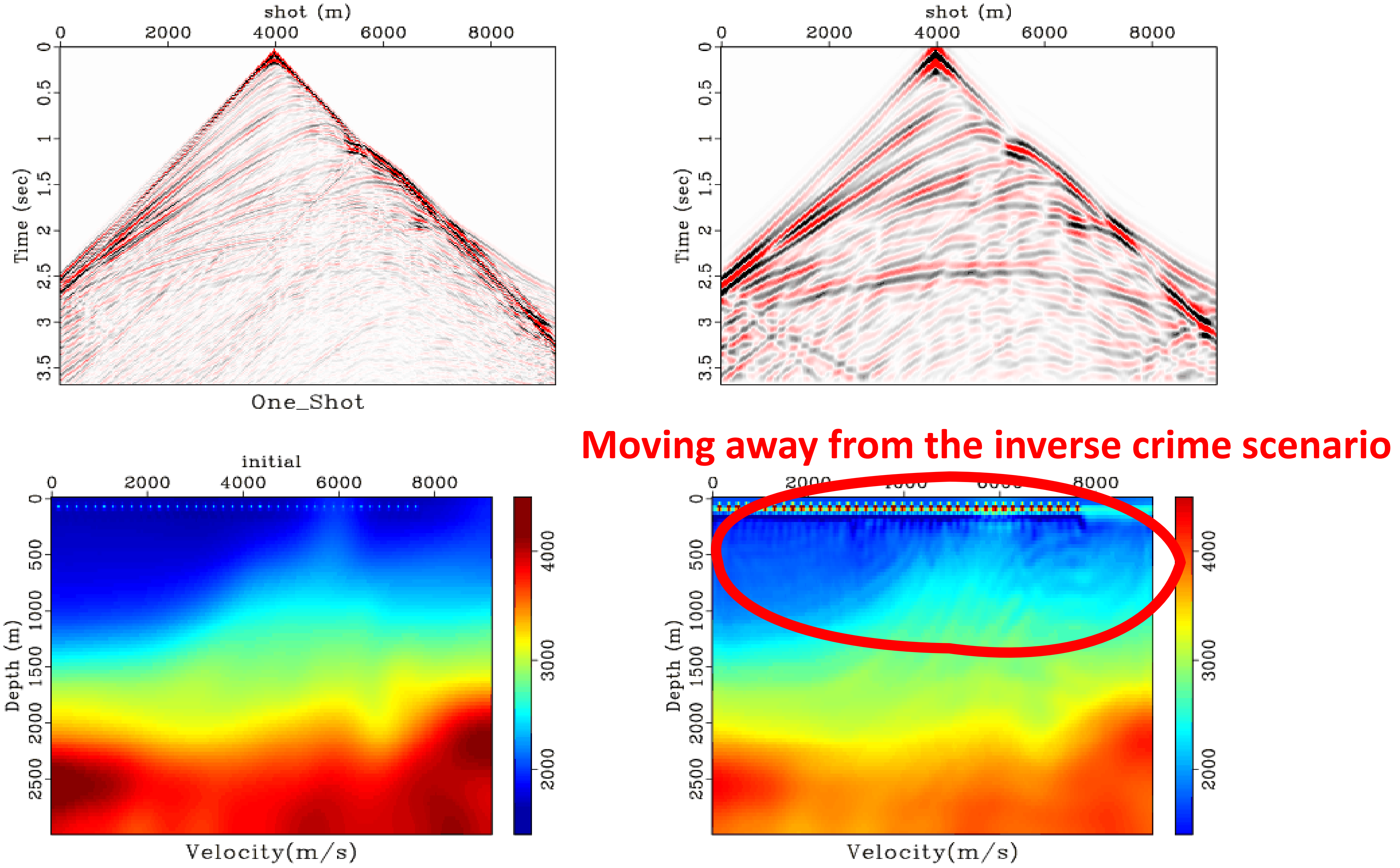
Dataflow I – Multigrid with band pass filter



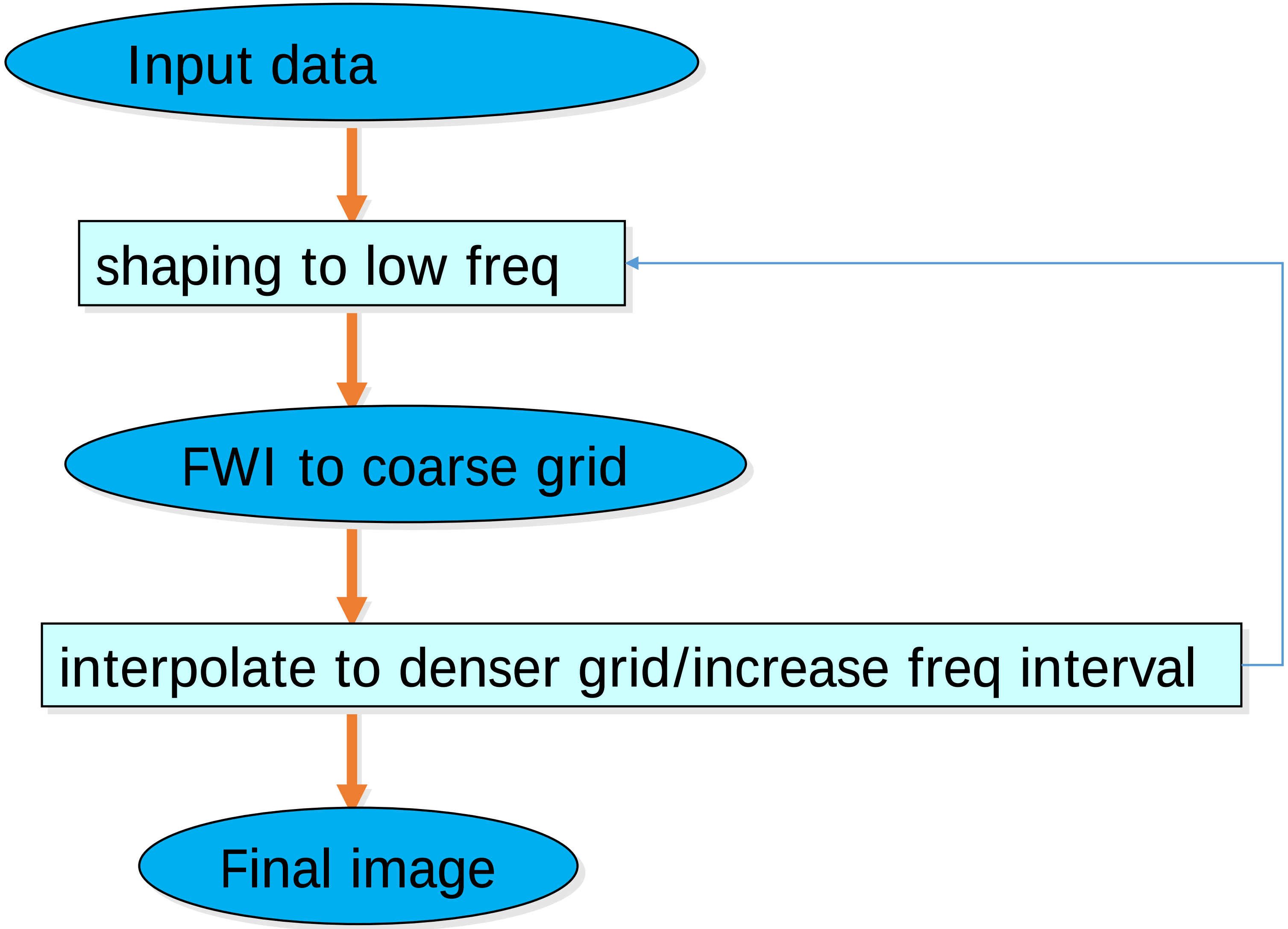
Multigrid from 0-8 Hz to 0-25 Hz and 32 m cells to 8 m cells



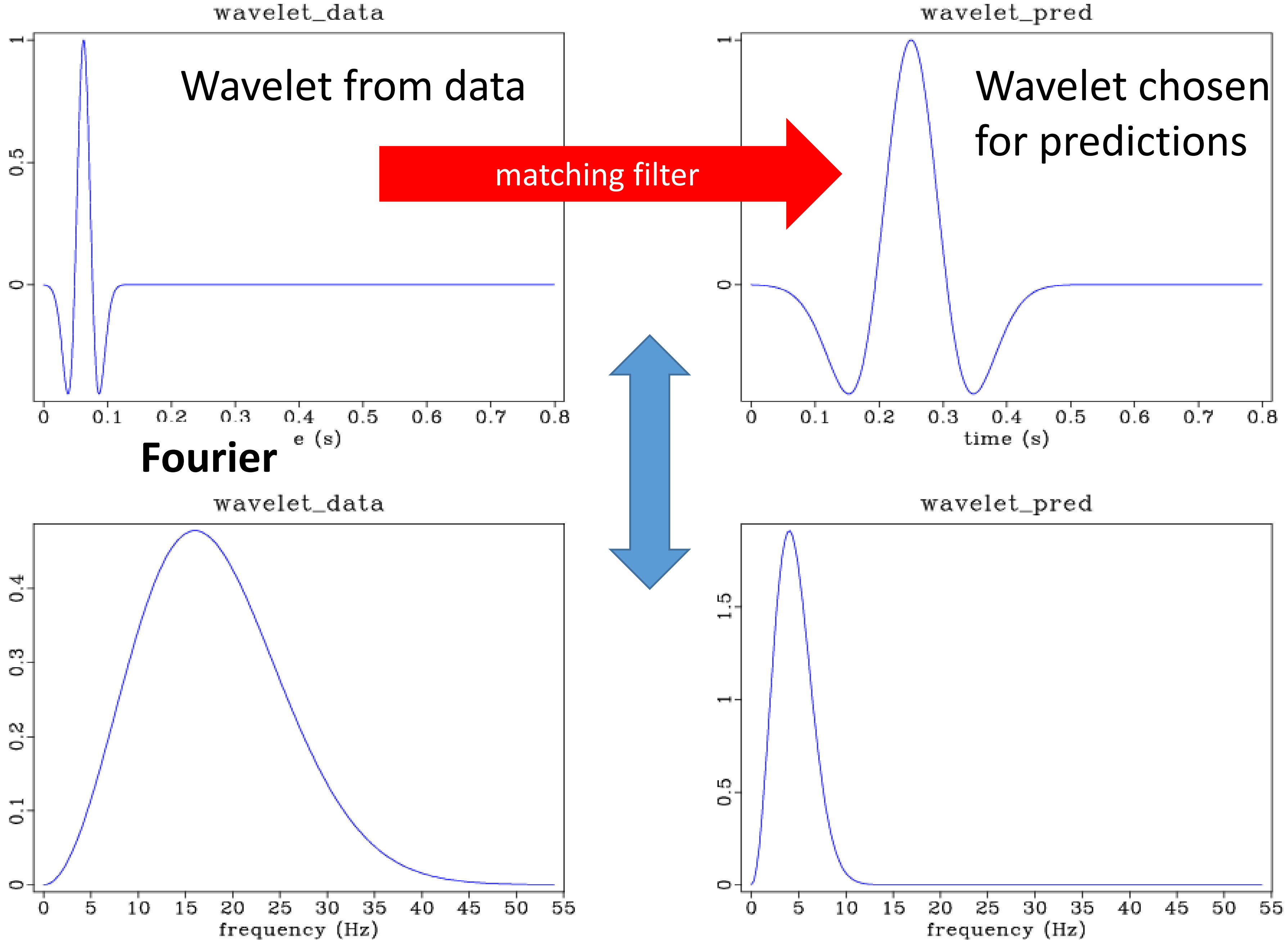
Result with simple band pass filter (0-8Hz from 0-25Hz)



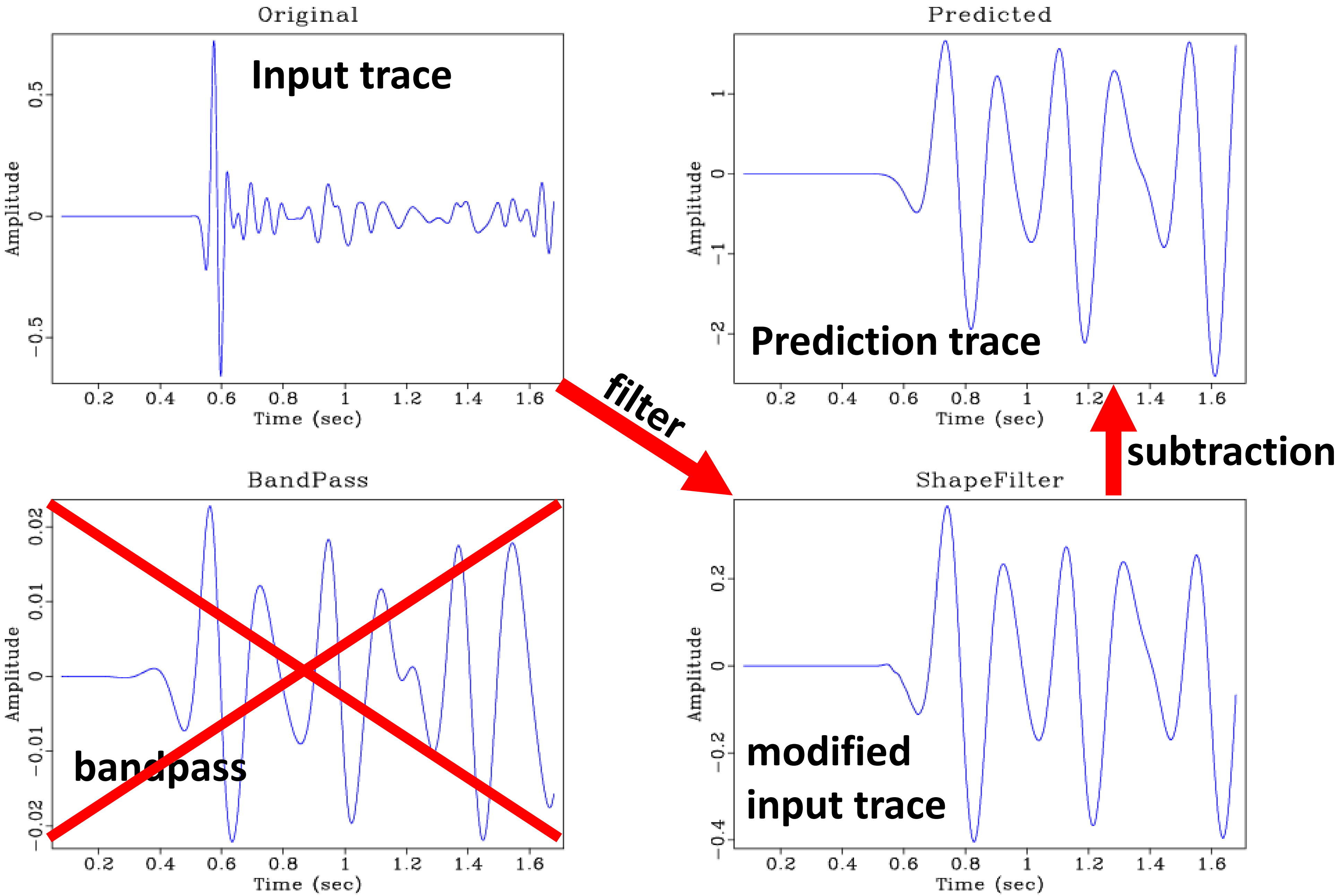
Dataflow II – Multigrid with shaping filter



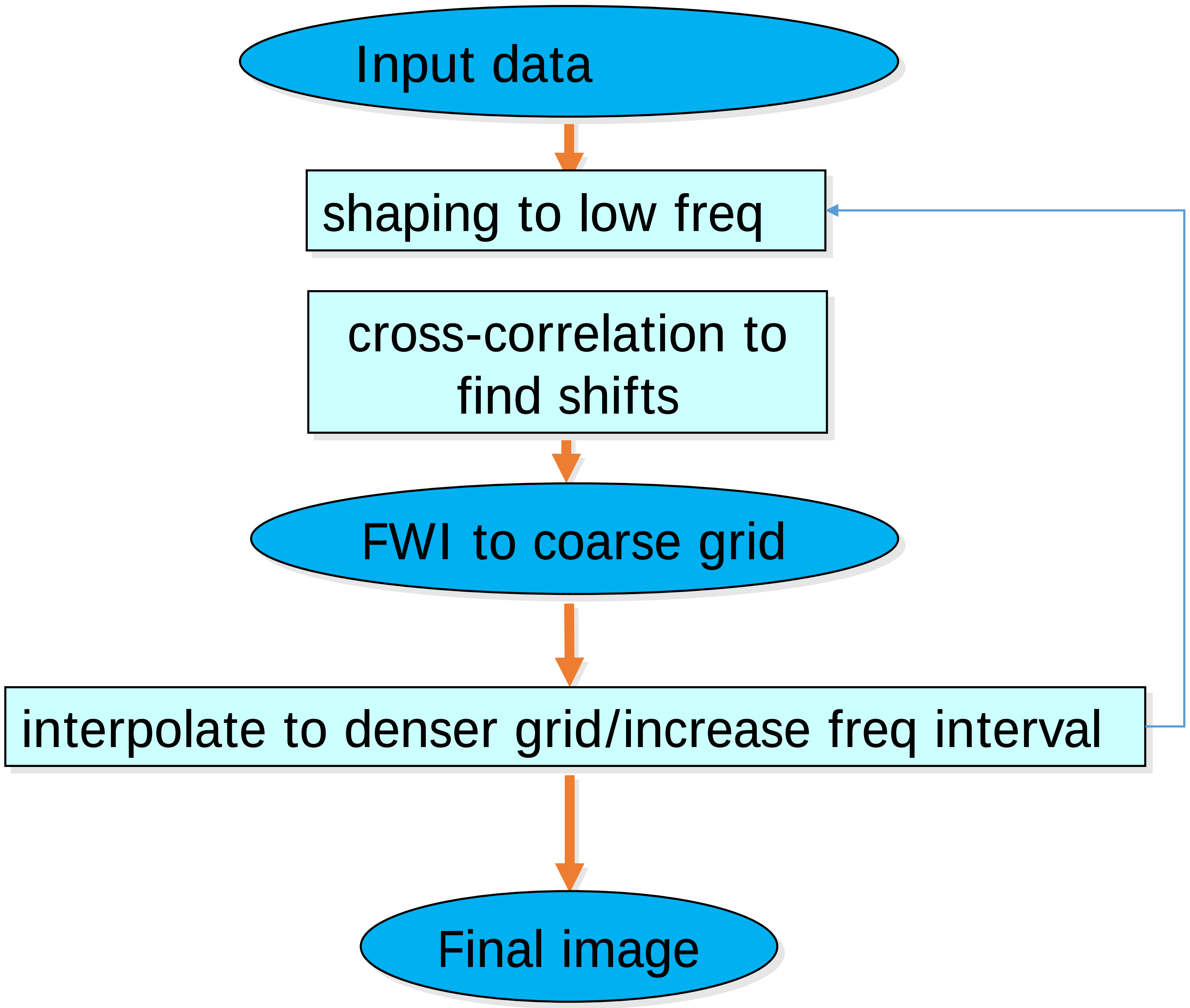
Shaping Filter



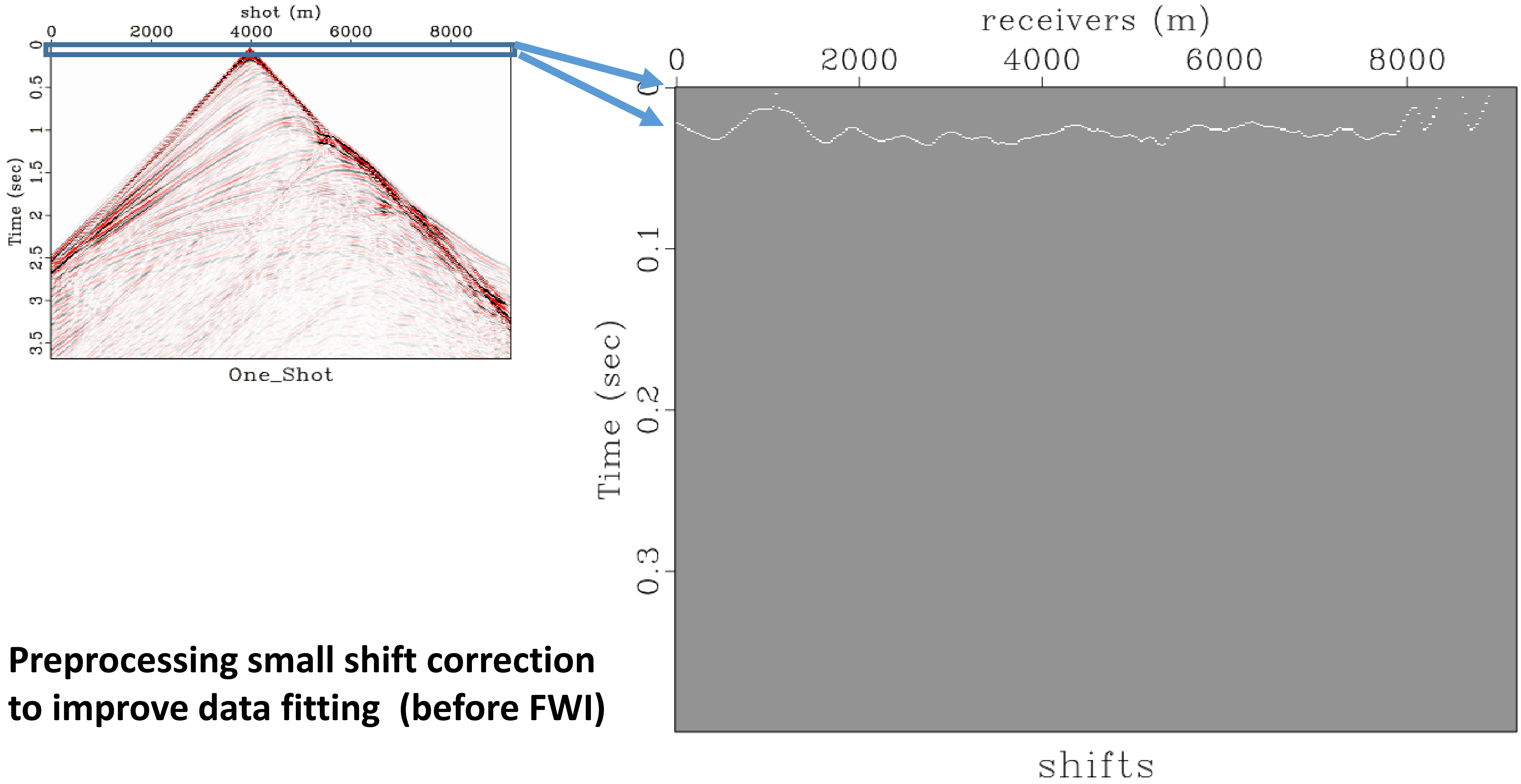
Wavelet from original, predicted, band pass filter, shape filter



Dataflow III – Multigrid with shaping filter and cross-correlation shifts.

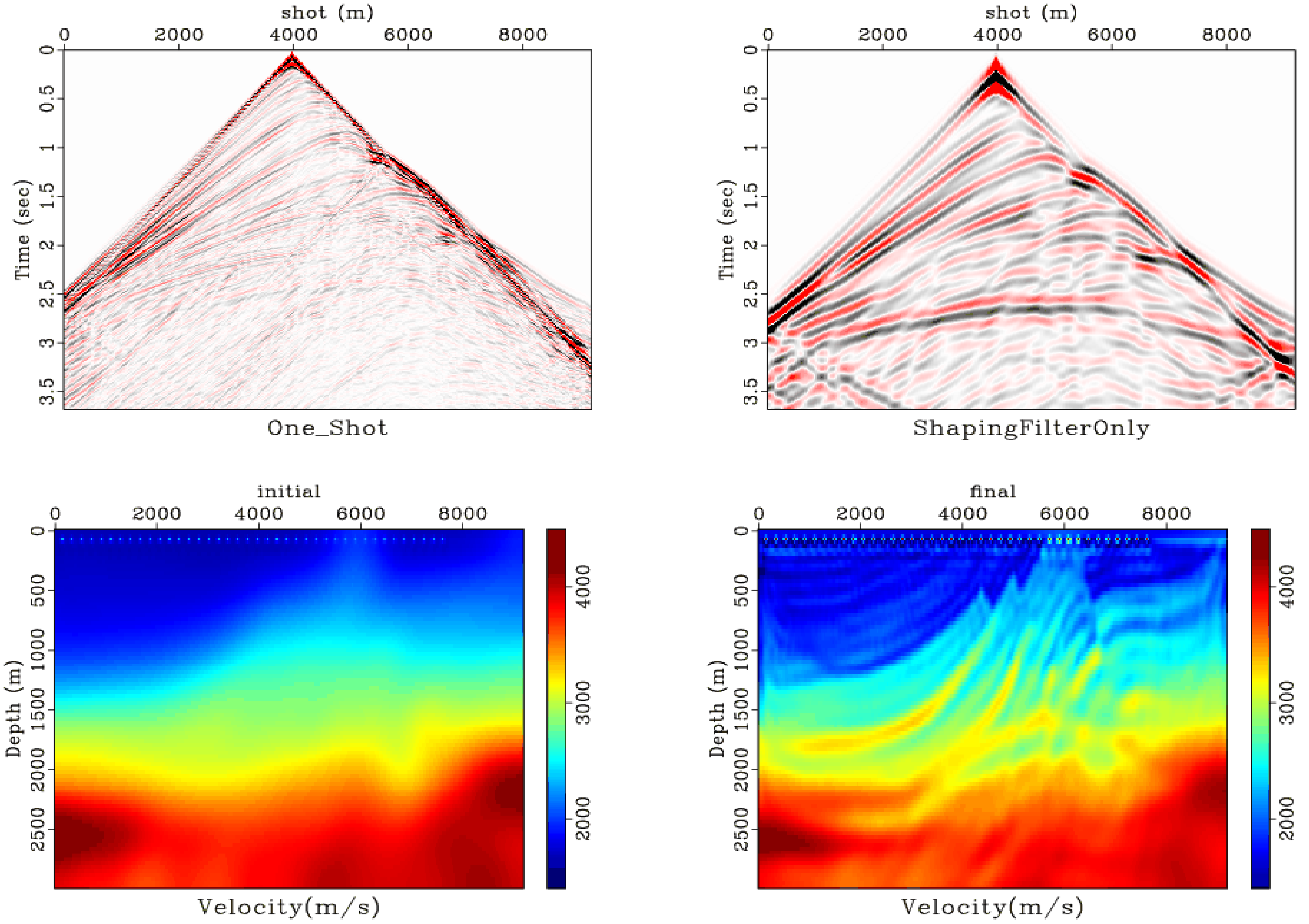


Cross-correlation shifts for one shot

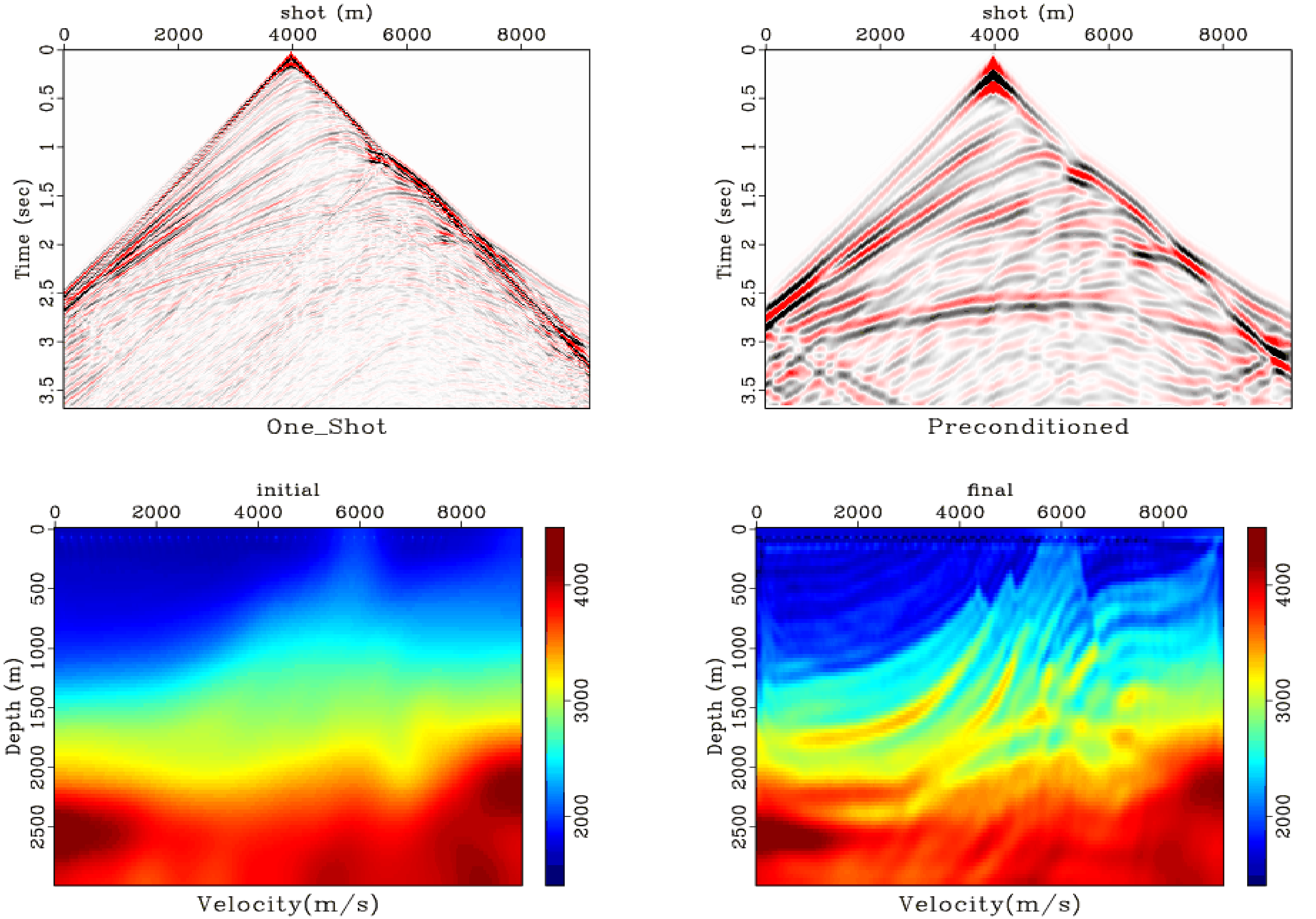


Preprocessing small shift correction to improve data fitting (before FWI)

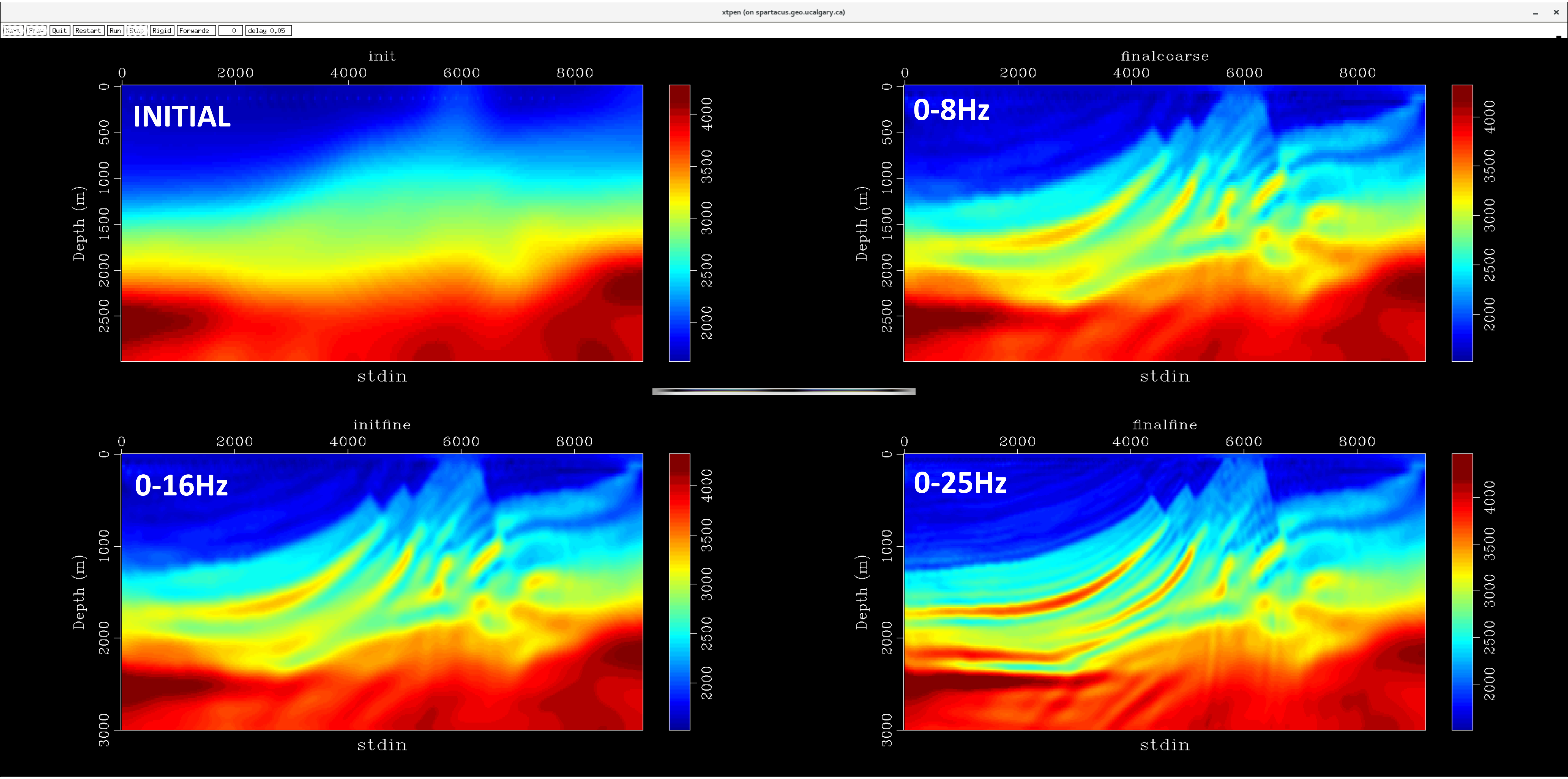
FWI (0-8Hz) from Input at (0-25Hz): shaping filter applied



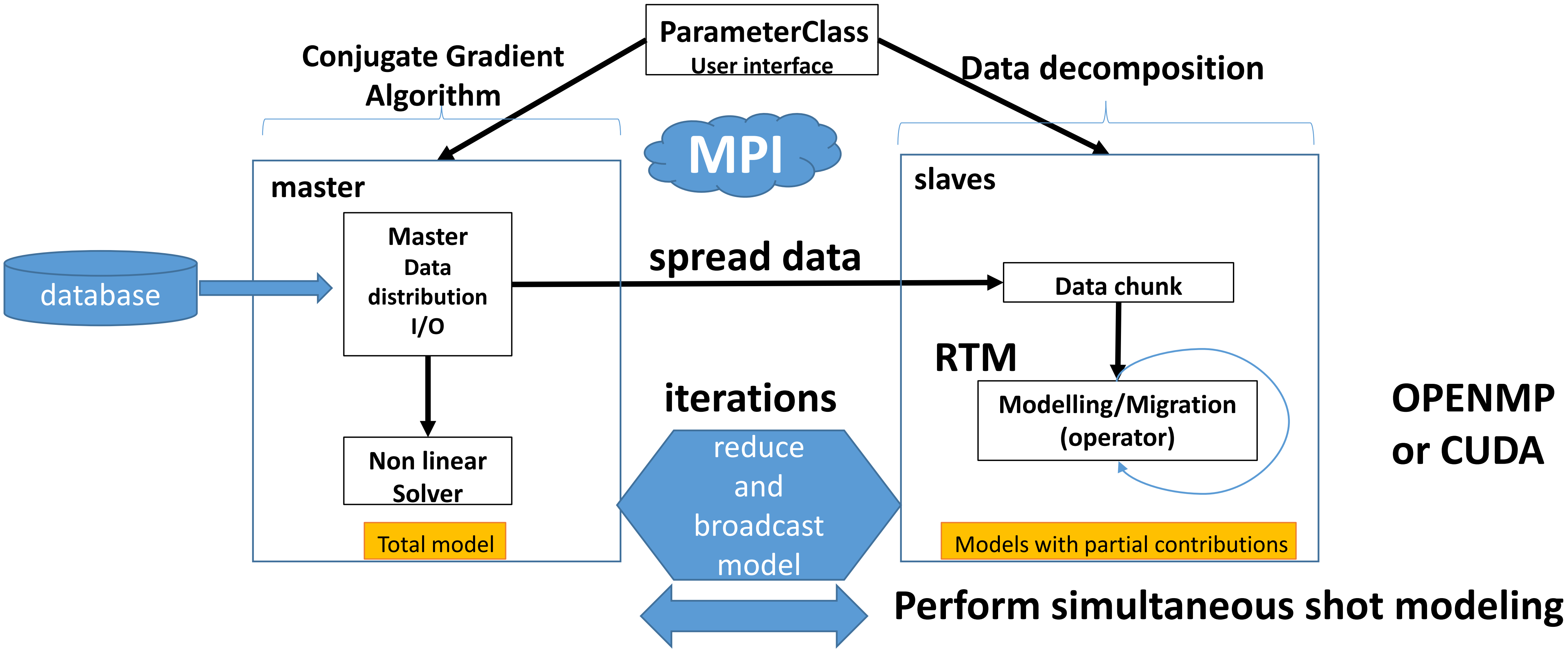
FWI (0-8Hz) from Input at (0-25Hz): shaping filter + xcorr-shift



Multigrid approach from 0-25Hz using grids of 32, 16 and 8m



Hybrid Parallel Implementation

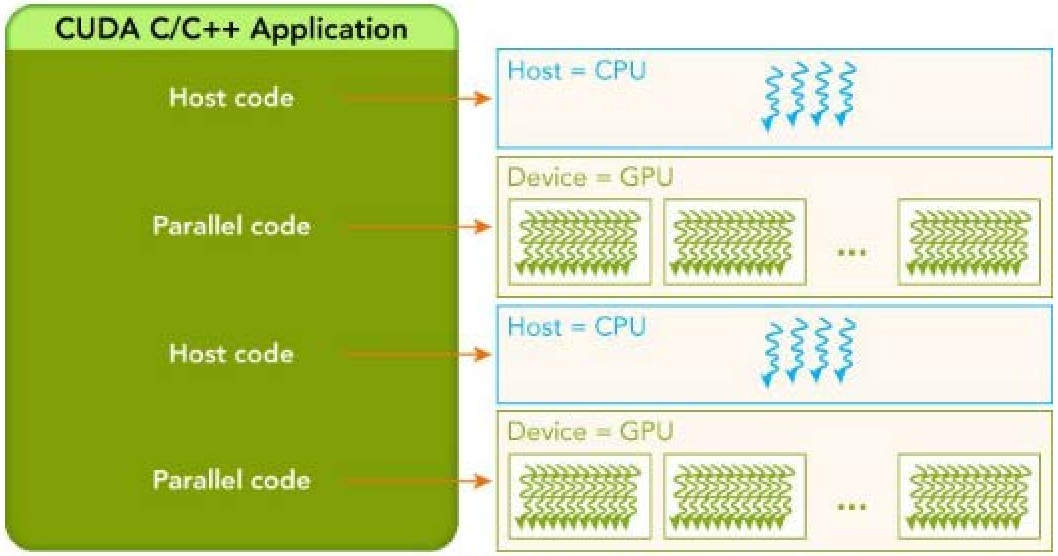


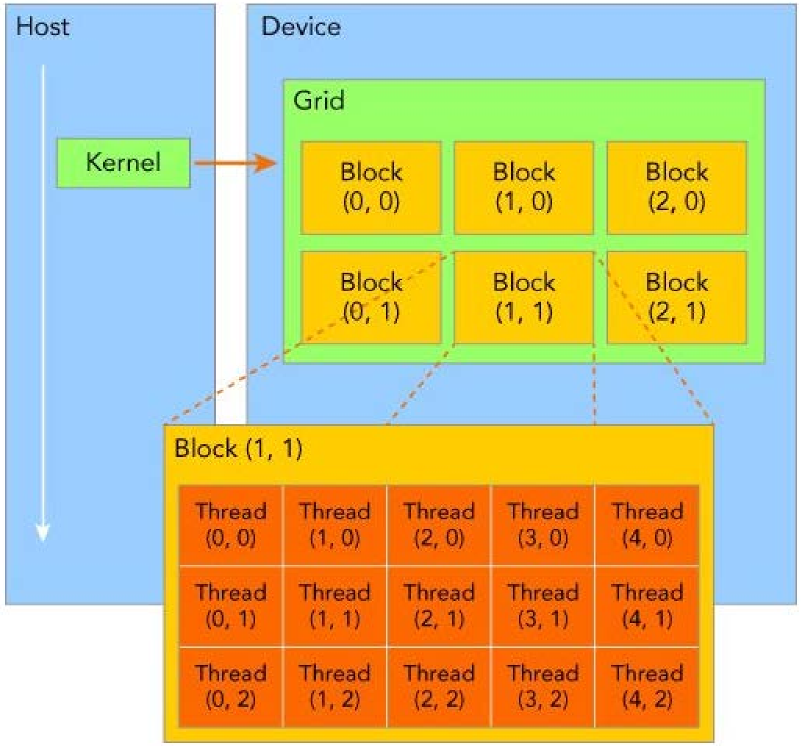
MPI-OPENMP hybrid model:
Small cluster 3 years old (25K US):
10 nodes, i7 6th generation, 8 threads,
128Gb per node

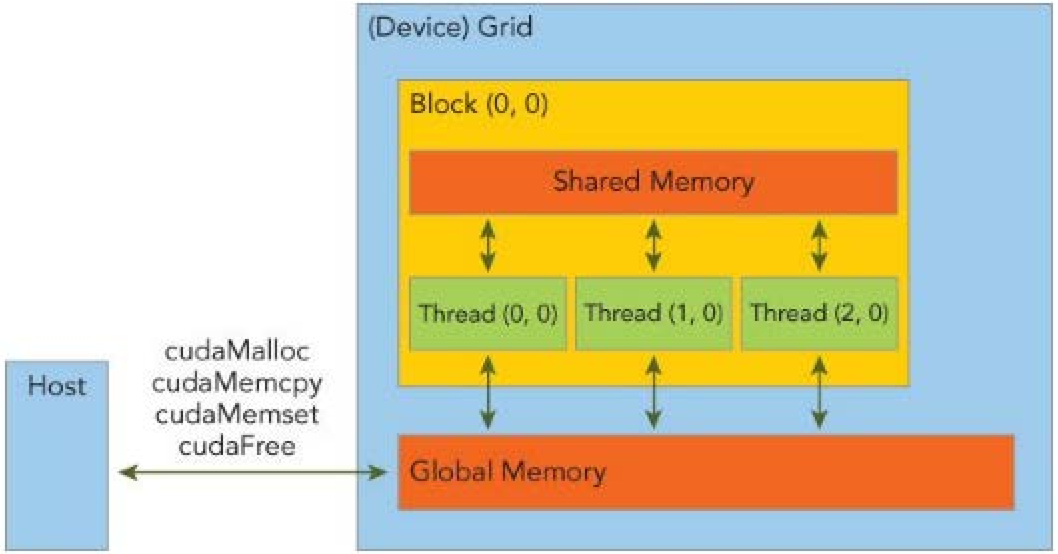
Table 1. Computation times

model size	cell size	time steps	nshots	iterations	time
96 x 288	32, 32	2800	40	20	196 secs
188 x 576	16, 16	2800	40	20	576 secs
376 x 1151	8, 8	4600	40	20	4481 secs

GPUs and programming in CUDA







Modeling times with OPENMP, OPENMPI and CUDA (same desktop)

Second Order:

- OPENMP, 12 threads, 10 shots → 18 sec
- CUDA, RTX2070, 10 shots → **3** sec

Fourth Order:

- OPENMP, 12 threads, 10 shots → 220 sec
- OPENMPI, 12 threads (1node=1thread), 10 shots → **60** sec
- CUDA, RTX2070, 10 shots → **3** sec

(gain of 5-100 times)

SUMMARY

When testing FWI is critical not to fall into the inverse crime.

Working with multigrids in the time domain helps to break the inverse crime circle because artifacts change when cell size changes. However this also requires to adapt the data or the predictions with some kind of filter.

Time domain FWI 2D or 3D is very efficient because it allows coarse grained parallelization with MPI (gain X 20 in Spartacus).

Frequency domain FWI, in 2D, can be made efficient with fine grained parallelization but it requires a cluster with fast communication across nodes.

GPUs allow very fast Time Domain finite differences but require careful programming with CUDA using shared memory (gain X 10)

Acknowledgments

Support:

- CREWES sponsors
- Canadian SEG (Chair in Exploration Geophysics)
- Natural Sciences and Engineering Research Council of Canada

special thanks to Penliang Yang, S. Keating, W. Pan and Sam Gray.

References

- **Time Domain FWI**

A graphics processing unit implementation of time-domain full-waveform inversion
P Yang, J Gao, B Wang - Geophysics, 2015 - library.seg.org

- **The importance of modeling for inversion**

Assumptions and goals for least squares migration, D Trad, 2020, Geophysical Prospecting