Large-scale seismic data interpolation in a parallel computing environment

Curt Da Silva

SLIM
University of British Columbia

Parallel matrix completion for missing source/receiver interpolation

Goals

Extend the previous implementation of spgLR to a parallel version

Handle very large scale data volumes stored across multiple nodes

Matrix completion

We use SPGLR to solve

$$\min_{\mathbf{L},\mathbf{R}} \quad \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right)$$
$$\text{such that} \quad \|\mathcal{A}(\mathbf{L}\mathbf{R}^T) - b\|_F \leq \sigma$$

by computing

$$v(\tau) = \min_{\mathbf{L},\mathbf{R}} \quad \|\mathcal{A}(\mathbf{L}\mathbf{R}^T) - b\|_F^2$$
$$\text{such that} \quad \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \leq \tau$$

and finding τ such that $v(\tau) = \sigma^2$

SPGLR

Basic operations of SPGLR

- compute objective, gradient
  - involves computing $\mathbf{X} = \mathbf{L}\mathbf{R}^T$, we'll come back to this

- project on to $\frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \leq \tau$ ball, inner products, norms
  - trivial to parallelize

- compute $v'(\tau)$, the derivative of the value function

$$v(\tau) = \min_{\mathbf{L},\mathbf{R}} \quad \|\mathcal{A}(\mathbf{L}\mathbf{R}^T) - b\|_F^2$$

$$\text{such that} \quad \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \leq \tau$$

Computing $v'(\tau)$

Normally involves computing the largest singular value of the data matrix

- can be done iteratively, but still expensive

Instead, we use a secant approximation

$$v'(\tau) \approx (v(\tau + h) - v(\tau))/h$$

for some small h

Computing $v'(\tau)$

Each evaluation of $v(\tau)$ involves solving

$$v(\tau) = \min_{\mathbf{L},\mathbf{R}} \quad \|\mathcal{A}(\mathbf{L}\mathbf{R}^T) - b\|_F^2$$

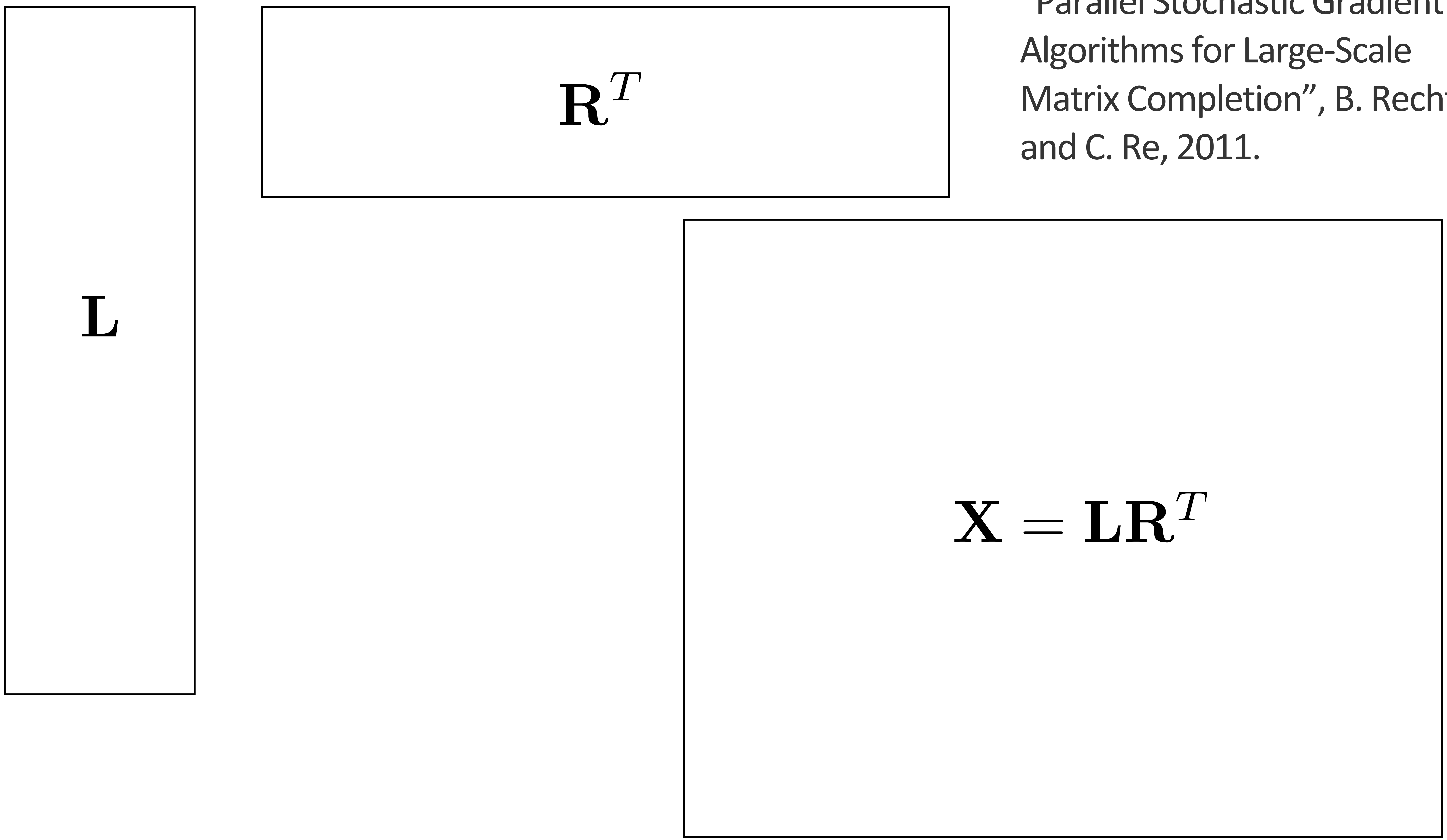
$$\text{such that} \quad \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \leq \tau$$

which we can already do in a distributed environment

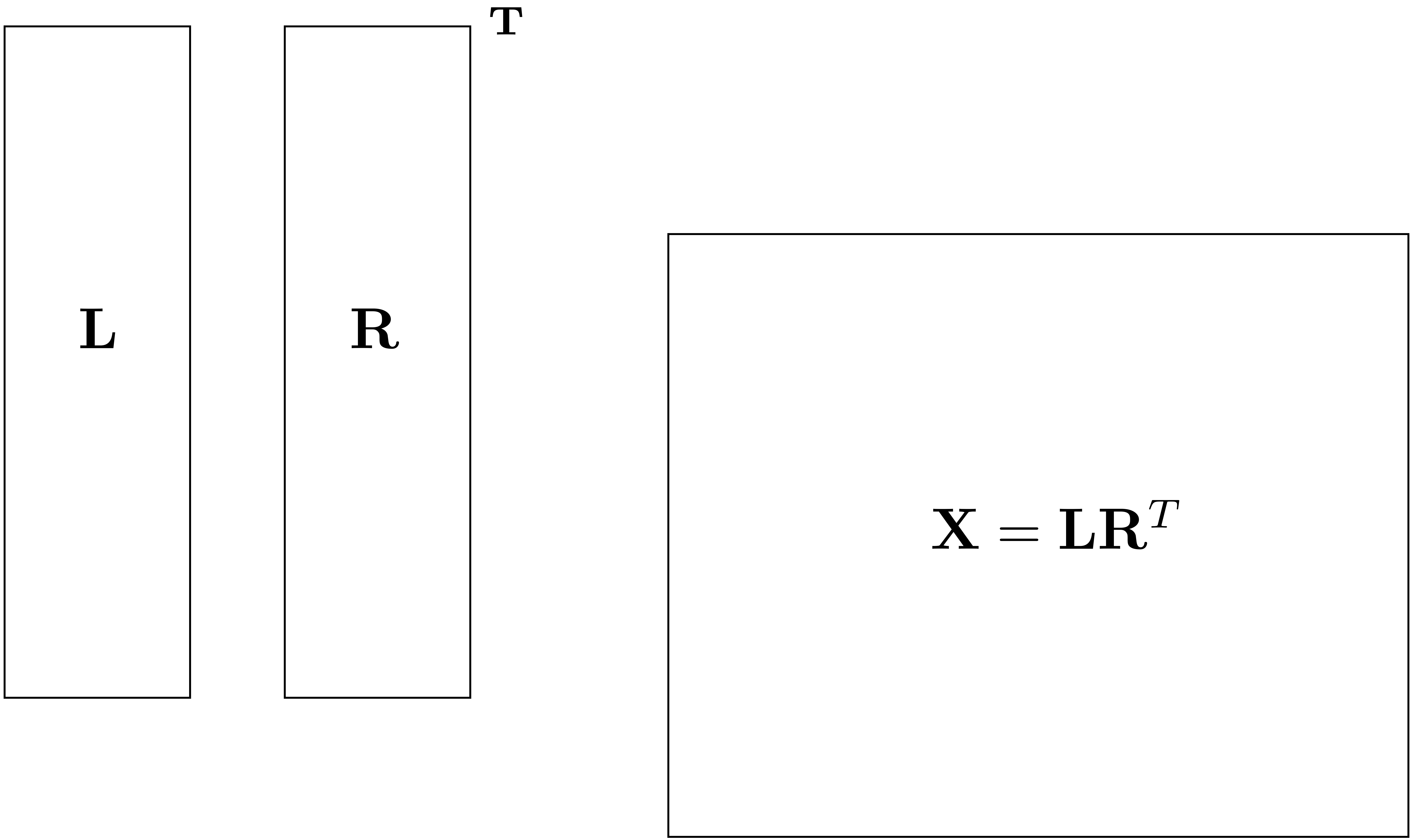
Using our approximation of $v'(\tau)$, we use Newton's method to update τ

LR parallel matrix multiplication

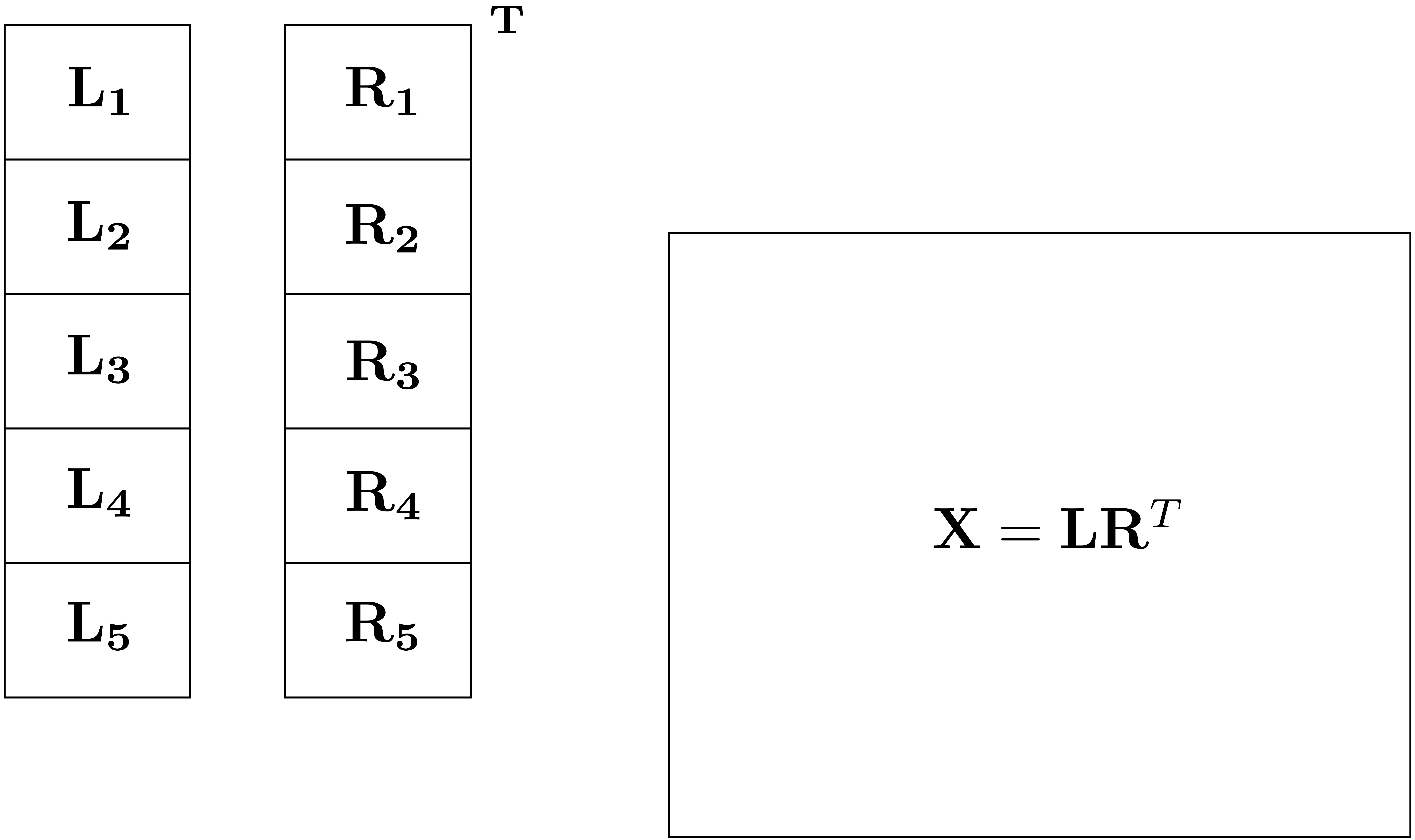
Cyclic block-permutations: “Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion”, B. Recht and C. Re, 2011.



LR parallel matrix multiplication



LR parallel matrix multiplication



LR parallel matrix multiplication

	$\mathbf{L}$	$\mathbf{R}^{\mathbf{T}}$
Worker 1	$\mathbf{L_1}$	$\mathbf{R_1}$
Worker 2	$\mathbf{L_2}$	$\mathbf{R_2}$
Worker 3	$\mathbf{L_3}$	$\mathbf{R_3}$
Worker 4	$\mathbf{L_4}$	$\mathbf{R_4}$
Worker 5	$\mathbf{L_5}$	$\mathbf{R_5}$

$$\mathbf{X} = \mathbf{L}\mathbf{R}^T$$

LR parallel matrix multiplication

	$\mathbf{L}$	$\mathbf{R}^{\mathbf{T}}$
Worker 1	$\mathbf{L_1}$	$\mathbf{R_1}$
Worker 2	$\mathbf{L_2}$	$\mathbf{R_2}$
Worker 3	$\mathbf{L_3}$	$\mathbf{R_3}$
Worker 4	$\mathbf{L_4}$	$\mathbf{R_4}$
Worker 5	$\mathbf{L_5}$	$\mathbf{R_5}$

$\mathbf{L_1R_1^T}$	$\mathbf{L_1R_2^T}$	$\mathbf{L_1R_3^T}$	$\mathbf{L_1R_4^T}$	$\mathbf{L_1R_5^T}$
$\mathbf{L_2R_1^T}$	$\mathbf{L_2R_2^T}$	$\mathbf{L_2R_3^T}$	$\mathbf{L_2R_4^T}$	$\mathbf{L_2R_5^T}$
$\mathbf{L_3R_1^T}$	$\mathbf{L_3R_2^T}$	$\mathbf{L_3R_3^T}$	$\mathbf{L_3R_4^T}$	$\mathbf{L_3R_5^T}$
$\mathbf{L_4R_1^T}$	$\mathbf{L_4R_2^T}$	$\mathbf{L_4R_3^T}$	$\mathbf{L_4R_4^T}$	$\mathbf{L_4R_5^T}$
$\mathbf{L_5R_1^T}$	$\mathbf{L_5R_2^T}$	$\mathbf{L_5R_3^T}$	$\mathbf{L_5R_4^T}$	$\mathbf{L_5R_5^T}$

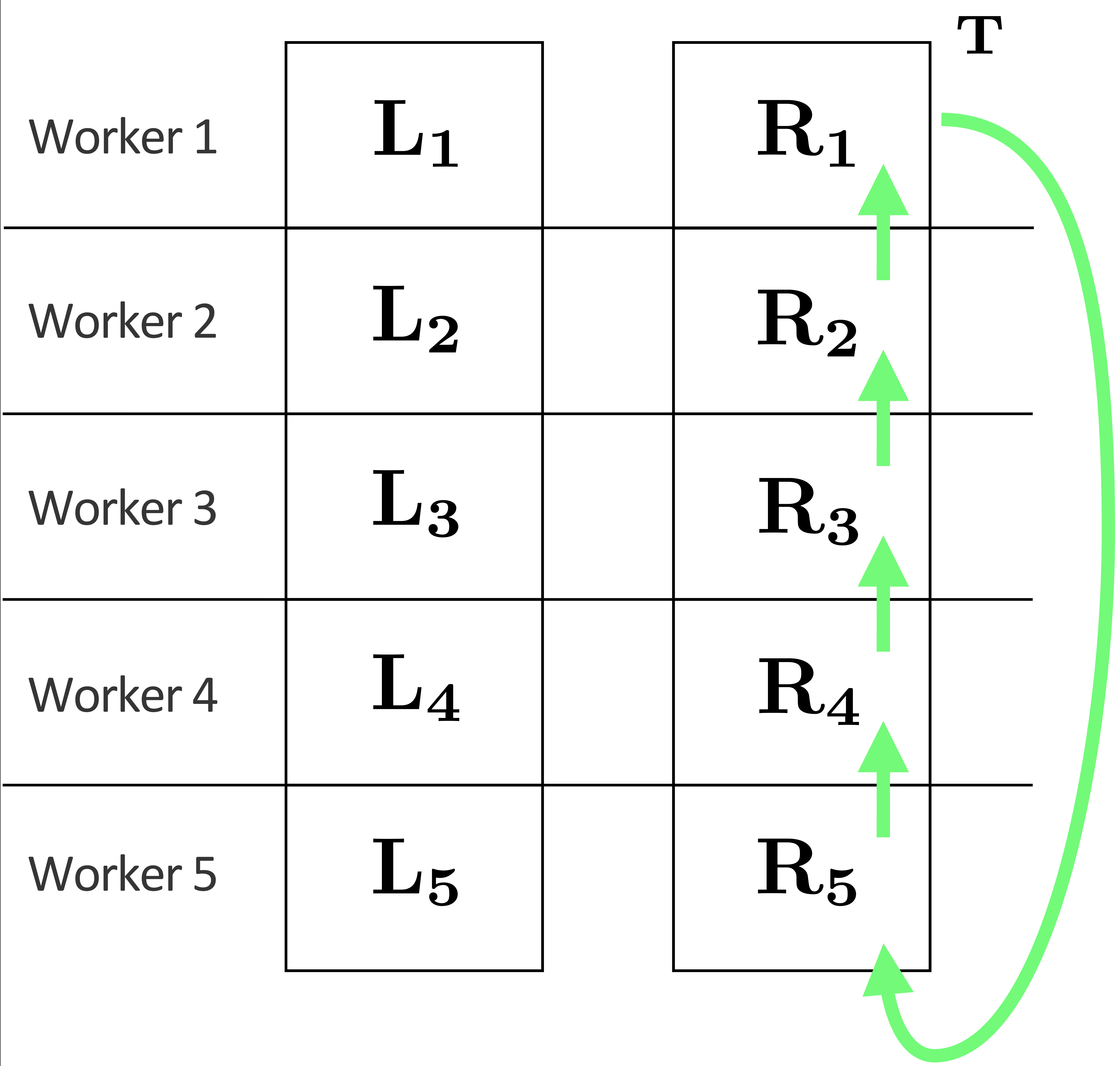
LR parallel matrix multiplication

- Process green blocks in parallel

	$\mathbf{L}$	$\mathbf{R}^{\mathbf{T}}$
Worker 1	$\mathbf{L_1}$	$\mathbf{R_1}$
Worker 2	$\mathbf{L_2}$	$\mathbf{R_2}$
Worker 3	$\mathbf{L_3}$	$\mathbf{R_3}$
Worker 4	$\mathbf{L_4}$	$\mathbf{R_4}$
Worker 5	$\mathbf{L_5}$	$\mathbf{R_5}$

$\mathbf{L_1R_1^T}$	$\mathbf{L_1R_2^T}$	$\mathbf{L_1R_3^T}$	$\mathbf{L_1R_4^T}$	$\mathbf{L_1R_5^T}$
$\mathbf{L_2R_1^T}$	$\mathbf{L_2R_2^T}$	$\mathbf{L_2R_3^T}$	$\mathbf{L_2R_4^T}$	$\mathbf{L_2R_5^T}$
$\mathbf{L_3R_1^T}$	$\mathbf{L_3R_2^T}$	$\mathbf{L_3R_3^T}$	$\mathbf{L_3R_4^T}$	$\mathbf{L_3R_5^T}$
$\mathbf{L_4R_1^T}$	$\mathbf{L_4R_2^T}$	$\mathbf{L_4R_3^T}$	$\mathbf{L_4R_4^T}$	$\mathbf{L_4R_5^T}$
$\mathbf{L_5R_1^T}$	$\mathbf{L_5R_2^T}$	$\mathbf{L_5R_3^T}$	$\mathbf{L_5R_4^T}$	$\mathbf{L_5R_5^T}$

LR parallel matrix multiplication



- Process green blocks in parallel
- Communicate R blocks to next worker

$\mathbf{L_1R_1^T}$	$\mathbf{L_1R_2^T}$	$\mathbf{L_1R_3^T}$	$\mathbf{L_1R_4^T}$	$\mathbf{L_1R_5^T}$
$\mathbf{L_2R_1^T}$	$\mathbf{L_2R_2^T}$	$\mathbf{L_2R_3^T}$	$\mathbf{L_2R_4^T}$	$\mathbf{L_2R_5^T}$
$\mathbf{L_3R_1^T}$	$\mathbf{L_3R_2^T}$	$\mathbf{L_3R_3^T}$	$\mathbf{L_3R_4^T}$	$\mathbf{L_3R_5^T}$
$\mathbf{L_4R_1^T}$	$\mathbf{L_4R_2^T}$	$\mathbf{L_4R_3^T}$	$\mathbf{L_4R_4^T}$	$\mathbf{L_4R_5^T}$
$\mathbf{L_5R_1^T}$	$\mathbf{L_5R_2^T}$	$\mathbf{L_5R_3^T}$	$\mathbf{L_5R_4^T}$	$\mathbf{L_5R_5^T}$

LR parallel matrix multiplication

- Process green blocks in parallel
- Communicate R blocks to next worker

		$\mathbf{T}$
Worker 1	$\mathbf{L}_1$	$\mathbf{R}_2$
Worker 2	$\mathbf{L}_2$	$\mathbf{R}_3$
Worker 3	$\mathbf{L}_3$	$\mathbf{R}_4$
Worker 4	$\mathbf{L}_4$	$\mathbf{R}_5$
Worker 5	$\mathbf{L}_5$	$\mathbf{R}_1$

$\mathbf{L}_1\mathbf{R}_1^T$	$\mathbf{L}_1\mathbf{R}_2^T$	$\mathbf{L}_1\mathbf{R}_3^T$	$\mathbf{L}_1\mathbf{R}_4^T$	$\mathbf{L}_1\mathbf{R}_5^T$
$\mathbf{L}_2\mathbf{R}_1^T$	$\mathbf{L}_2\mathbf{R}_2^T$	$\mathbf{L}_2\mathbf{R}_3^T$	$\mathbf{L}_2\mathbf{R}_4^T$	$\mathbf{L}_2\mathbf{R}_5^T$
$\mathbf{L}_3\mathbf{R}_1^T$	$\mathbf{L}_3\mathbf{R}_2^T$	$\mathbf{L}_3\mathbf{R}_3^T$	$\mathbf{L}_3\mathbf{R}_4^T$	$\mathbf{L}_3\mathbf{R}_5^T$
$\mathbf{L}_4\mathbf{R}_1^T$	$\mathbf{L}_4\mathbf{R}_2^T$	$\mathbf{L}_4\mathbf{R}_3^T$	$\mathbf{L}_4\mathbf{R}_4^T$	$\mathbf{L}_4\mathbf{R}_5^T$
$\mathbf{L}_5\mathbf{R}_1^T$	$\mathbf{L}_5\mathbf{R}_2^T$	$\mathbf{L}_5\mathbf{R}_3^T$	$\mathbf{L}_5\mathbf{R}_4^T$	$\mathbf{L}_5\mathbf{R}_5^T$

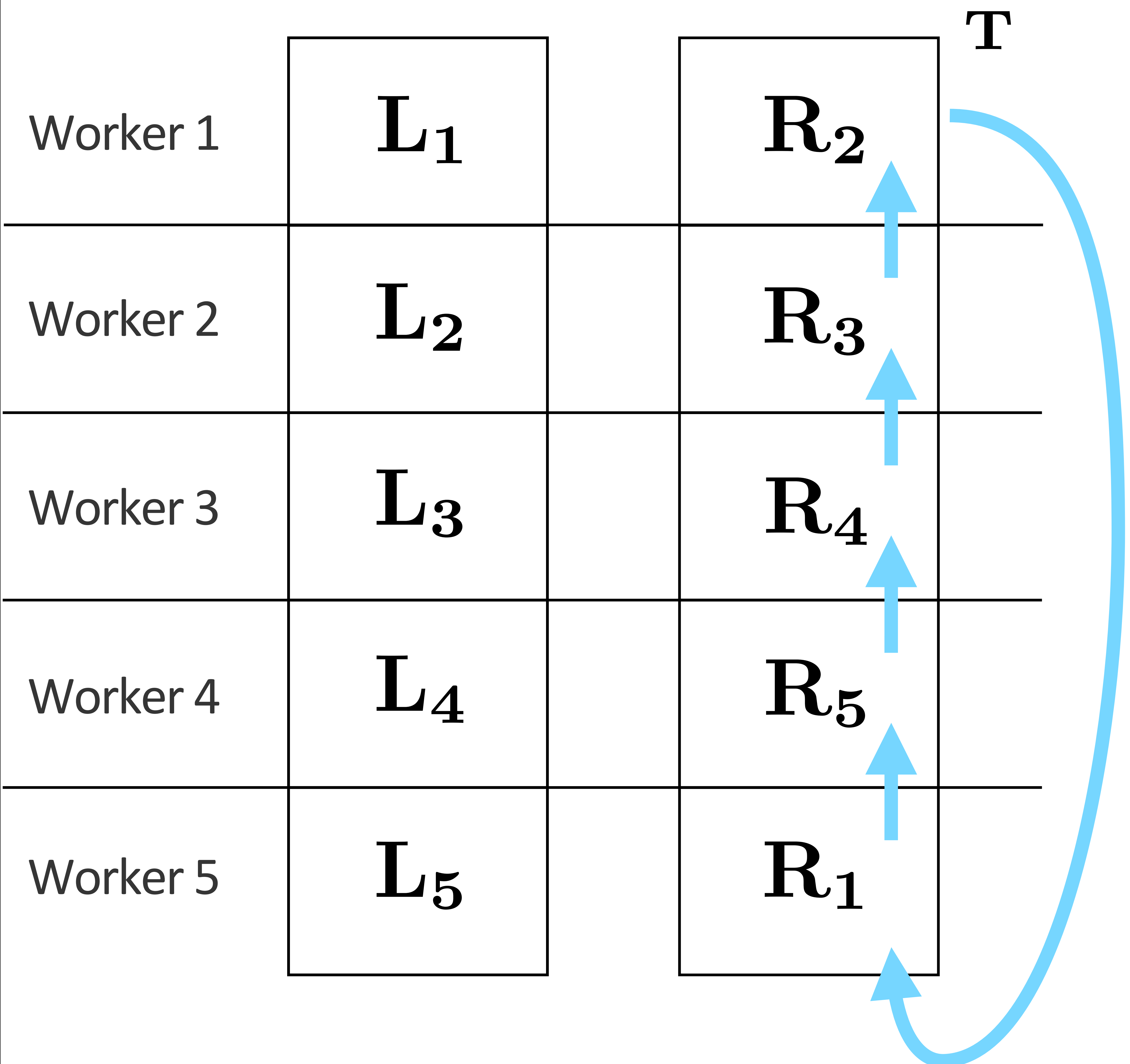
LR parallel matrix multiplication

- Process green blocks in parallel
- Communicate R blocks to next worker
- Process blue blocks in parallel

	$\mathbf{L}$	$\mathbf{R}^{\mathbf{T}}$
Worker 1	$\mathbf{L}_1$	$\mathbf{R}_2$
Worker 2	$\mathbf{L}_2$	$\mathbf{R}_3$
Worker 3	$\mathbf{L}_3$	$\mathbf{R}_4$
Worker 4	$\mathbf{L}_4$	$\mathbf{R}_5$
Worker 5	$\mathbf{L}_5$	$\mathbf{R}_1$

$\mathbf{L}_1\mathbf{R}_1^T$	$\mathbf{L}_1\mathbf{R}_2^T$	$\mathbf{L}_1\mathbf{R}_3^T$	$\mathbf{L}_1\mathbf{R}_4^T$	$\mathbf{L}_1\mathbf{R}_5^T$
$\mathbf{L}_2\mathbf{R}_1^T$	$\mathbf{L}_2\mathbf{R}_2^T$	$\mathbf{L}_2\mathbf{R}_3^T$	$\mathbf{L}_2\mathbf{R}_4^T$	$\mathbf{L}_2\mathbf{R}_5^T$
$\mathbf{L}_3\mathbf{R}_1^T$	$\mathbf{L}_3\mathbf{R}_2^T$	$\mathbf{L}_3\mathbf{R}_3^T$	$\mathbf{L}_3\mathbf{R}_4^T$	$\mathbf{L}_3\mathbf{R}_5^T$
$\mathbf{L}_4\mathbf{R}_1^T$	$\mathbf{L}_4\mathbf{R}_2^T$	$\mathbf{L}_4\mathbf{R}_3^T$	$\mathbf{L}_4\mathbf{R}_4^T$	$\mathbf{L}_4\mathbf{R}_5^T$
$\mathbf{L}_5\mathbf{R}_1^T$	$\mathbf{L}_5\mathbf{R}_2^T$	$\mathbf{L}_5\mathbf{R}_3^T$	$\mathbf{L}_5\mathbf{R}_4^T$	$\mathbf{L}_5\mathbf{R}_5^T$

LR parallel matrix multiplication



- Process green blocks in parallel
- Communicate R blocks to next worker
- Process blue blocks in parallel
- Communicate R blocks to next worker

$\mathbf{L_1R_1^T}$	$\mathbf{L_1R_2^T}$	$\mathbf{L_1R_3^T}$	$\mathbf{L_1R_4^T}$	$\mathbf{L_1R_5^T}$
$\mathbf{L_2R_1^T}$	$\mathbf{L_2R_2^T}$	$\mathbf{L_2R_3^T}$	$\mathbf{L_2R_4^T}$	$\mathbf{L_2R_5^T}$
$\mathbf{L_3R_1^T}$	$\mathbf{L_3R_2^T}$	$\mathbf{L_3R_3^T}$	$\mathbf{L_3R_4^T}$	$\mathbf{L_3R_5^T}$
$\mathbf{L_4R_1^T}$	$\mathbf{L_4R_2^T}$	$\mathbf{L_4R_3^T}$	$\mathbf{L_4R_4^T}$	$\mathbf{L_4R_5^T}$
$\mathbf{L_5R_1^T}$	$\mathbf{L_5R_2^T}$	$\mathbf{L_5R_3^T}$	$\mathbf{L_5R_4^T}$	$\mathbf{L_5R_5^T}$

LR parallel matrix multiplication

- Process green blocks in parallel
- Communicate R blocks to next worker
- Process blue blocks in parallel
- Communicate R blocks to next worker

		$\mathbf{R}^T$
Worker 1	$\mathbf{L}_1$	$\mathbf{R}_3$
Worker 2	$\mathbf{L}_2$	$\mathbf{R}_4$
Worker 3	$\mathbf{L}_3$	$\mathbf{R}_5$
Worker 4	$\mathbf{L}_4$	$\mathbf{R}_1$
Worker 5	$\mathbf{L}_5$	$\mathbf{R}_2$

$\mathbf{L}_1\mathbf{R}_1^T$	$\mathbf{L}_1\mathbf{R}_2^T$	$\mathbf{L}_1\mathbf{R}_3^T$	$\mathbf{L}_1\mathbf{R}_4^T$	$\mathbf{L}_1\mathbf{R}_5^T$
$\mathbf{L}_2\mathbf{R}_1^T$	$\mathbf{L}_2\mathbf{R}_2^T$	$\mathbf{L}_2\mathbf{R}_3^T$	$\mathbf{L}_2\mathbf{R}_4^T$	$\mathbf{L}_2\mathbf{R}_5^T$
$\mathbf{L}_3\mathbf{R}_1^T$	$\mathbf{L}_3\mathbf{R}_2^T$	$\mathbf{L}_3\mathbf{R}_3^T$	$\mathbf{L}_3\mathbf{R}_4^T$	$\mathbf{L}_3\mathbf{R}_5^T$
$\mathbf{L}_4\mathbf{R}_1^T$	$\mathbf{L}_4\mathbf{R}_2^T$	$\mathbf{L}_4\mathbf{R}_3^T$	$\mathbf{L}_4\mathbf{R}_4^T$	$\mathbf{L}_4\mathbf{R}_5^T$
$\mathbf{L}_5\mathbf{R}_1^T$	$\mathbf{L}_5\mathbf{R}_2^T$	$\mathbf{L}_5\mathbf{R}_3^T$	$\mathbf{L}_5\mathbf{R}_4^T$	$\mathbf{L}_5\mathbf{R}_5^T$

LR parallel matrix multiplication

	L	R^T
Worker 1	L_1	R_3
Worker 2	L_2	R_4
Worker 3	L_3	R_5
Worker 4	L_4	R_1
Worker 5	L_5	R_2

- Process green blocks in parallel
- Communicate R blocks to next worker
- Process blue blocks in parallel
- Communicate R blocks to next worker
- Repeat

$L_1R_1^T$	$L_1R_2^T$	$L_1R_3^T$	$L_1R_4^T$	$L_1R_5^T$
$L_2R_1^T$	$L_2R_2^T$	$L_2R_3^T$	$L_2R_4^T$	$L_2R_5^T$
$L_3R_1^T$	$L_3R_2^T$	$L_3R_3^T$	$L_3R_4^T$	$L_3R_5^T$
$L_4R_1^T$	$L_4R_2^T$	$L_4R_3^T$	$L_4R_4^T$	$L_4R_5^T$
$L_5R_1^T$	$L_5R_2^T$	$L_5R_3^T$	$L_5R_4^T$	$L_5R_5^T$

SLIM

Costs

$\mathbf{X}$ - $m \times n$ stored on p workers, $\mathbf{L}, \mathbf{R}$ each of rank k

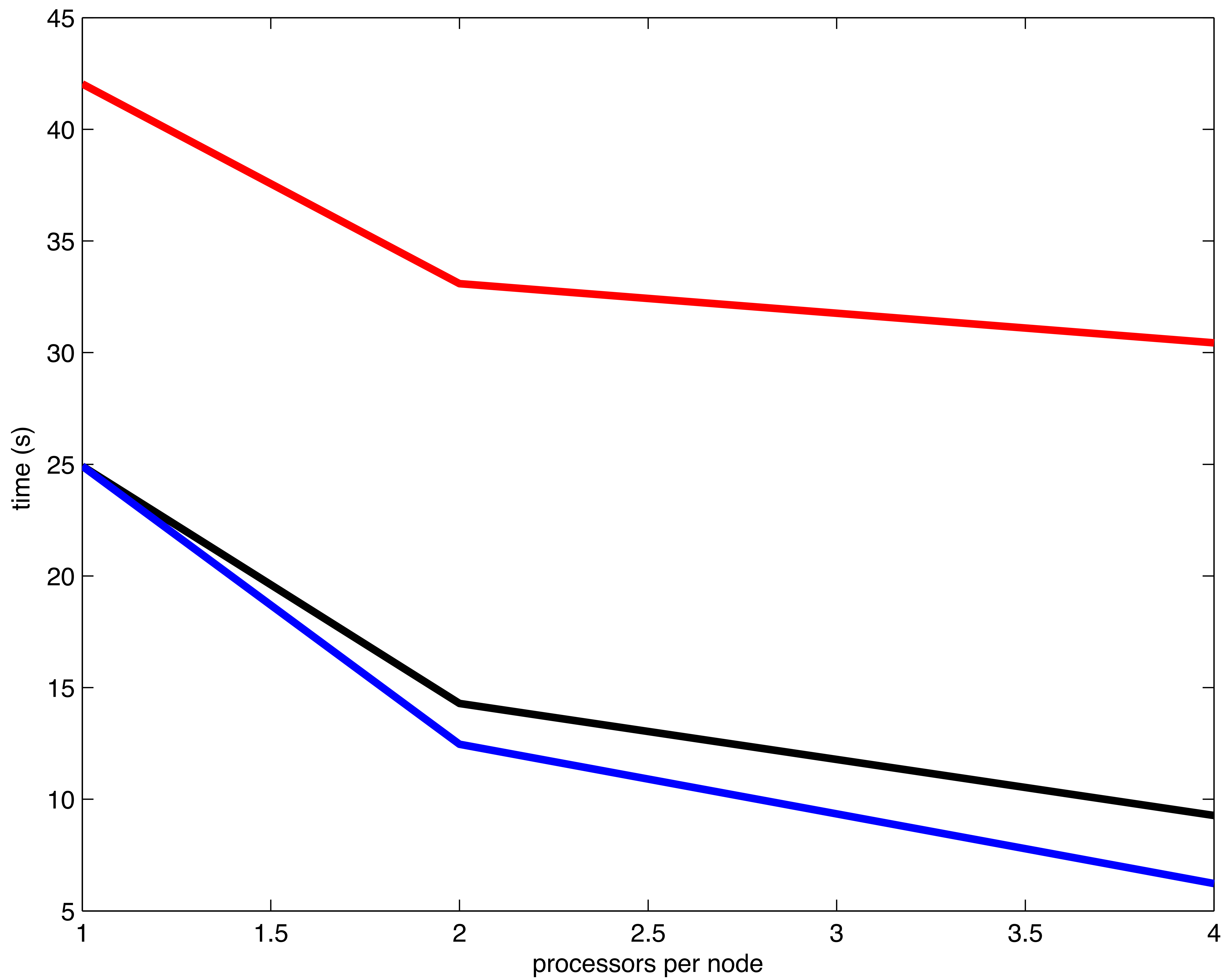
Communication per cycle : $O(nk)$ values

Total communication: $O(nkp) \ll O(mn)$ when $k, p \ll \sqrt{m}$

- much less than the full matrix

Only small submatrices in memory at any given time, *not* the full matrix

Performance scaling - evaluation of objective



Black - spgLR
Red - Naive Matlab
Blue - 1/#processors

Initial guess

Naive initialization for the algorithm

Let $\mathcal{A}^*b = \mathbf{U}\mathbf{S}\mathbf{V}^T$ be the rank k SVD of the zero-filled data

- $\mathbf{L} = \mathbf{U}\mathbf{S}^{-1/2}, \mathbf{R} = \mathbf{V}\mathbf{S}^{-1/2}$

- $\mathcal{A}^*b$ is an enormous, distributed matrix, SVD costly computationally, "overkill" as an initial guess

- can be shown to be "close" to the unknown matrix in Frobenius norm, so in some sense an optimal initial guess

Initial guess

Naive initialization for the algorithm

Let $\mathbf{L}, \mathbf{R}$ be initialized as Gaussian random matrices, appropriately scaled

- $\mathbf{L}\mathbf{R}^T$ is, in general, far from the true matrix $\mathbf{X}$ in Frobenius norm
- requires more iterations to decrease error below prescribed tolerance
- alternatively, for fixed number of iterations, error is higher using this initialization compared to one based on the data

Initial guess

Want to approximate the k-rank SVD without having to compute it exactly

- equivalent to $\min_{\mathbf{Q}\in\mathbb{R}^{m\times k}} \quad \|\mathbf{X}-\mathbf{Q}\mathbf{Q}^T\mathbf{X}\|_F^2$

$$\text{such that} \quad \mathbf{Q}^T\mathbf{Q}=\mathbf{I}$$

Approximate solution:

- Let $\mathbf{Y}=\mathbf{X}\Omega$ where Ω is a $n\times k$ Gaussian matrix

- Set $\mathbf{L}=\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2}, \mathbf{R}=\mathbf{X}^T\mathbf{L}$

Initial guess

Approximate solution: $\mathbf{Y} = \mathbf{X}\Omega \;\; \mathbf{L} = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2} \;\; \mathbf{R} = \mathbf{X}^T\mathbf{L}$

When $\mathbf{X} = \mathcal{A}^* b$ is the zero-filled data, initialization only involves

- matrix-matrix multiplications - efficient in a distributed environment
- eigenvalue decomposition of a $k \times k$ matrix - easy when k is small
- much closer to the data in Frobenius norm than random noise initialization

Results

BG Data set

Single frequency slice, real part

68 x 68 source grid at 150m spacing

401 x 401 receiver grid at 50m spacing

SLIM

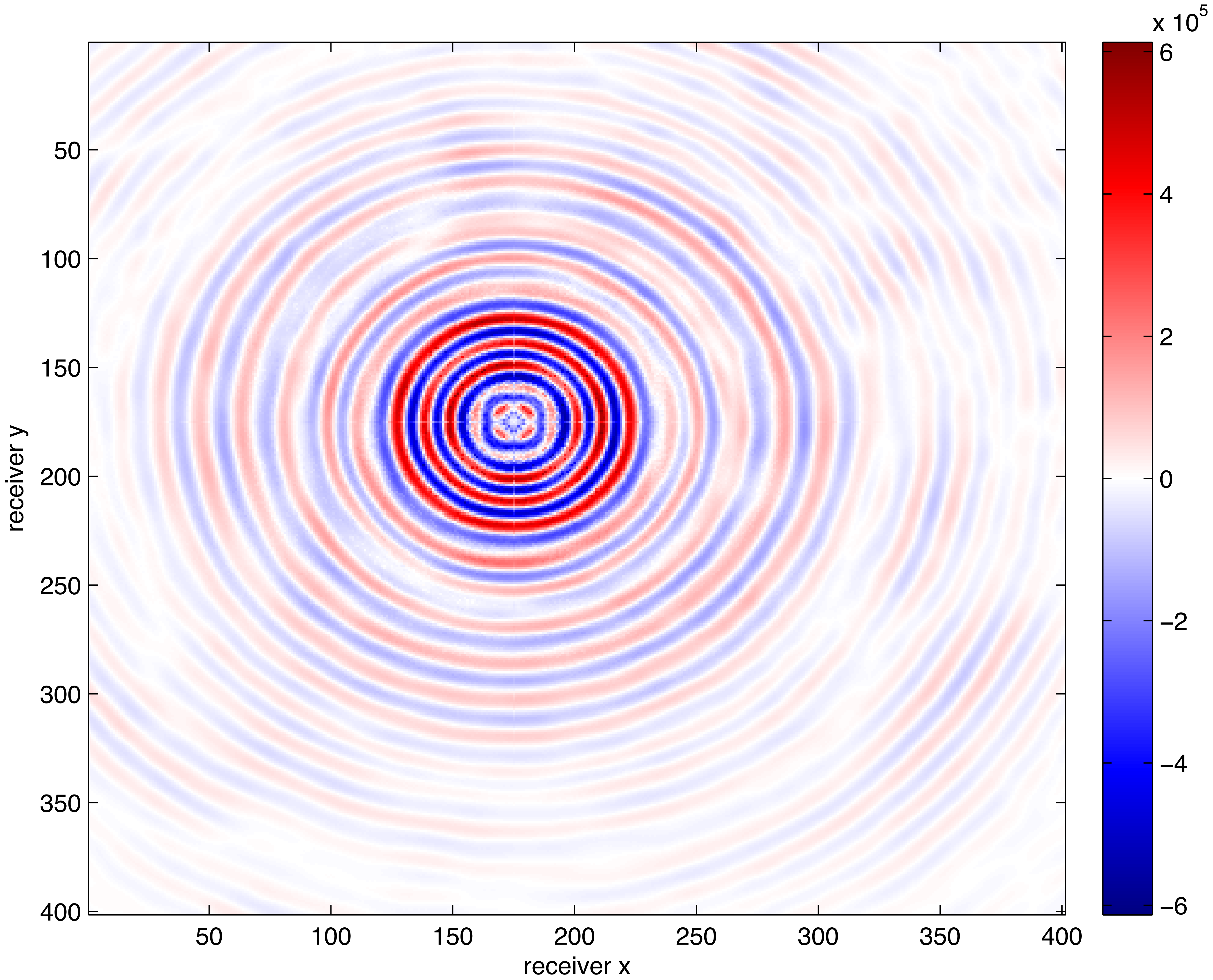
BG Data set

500 iterations of SPGLR

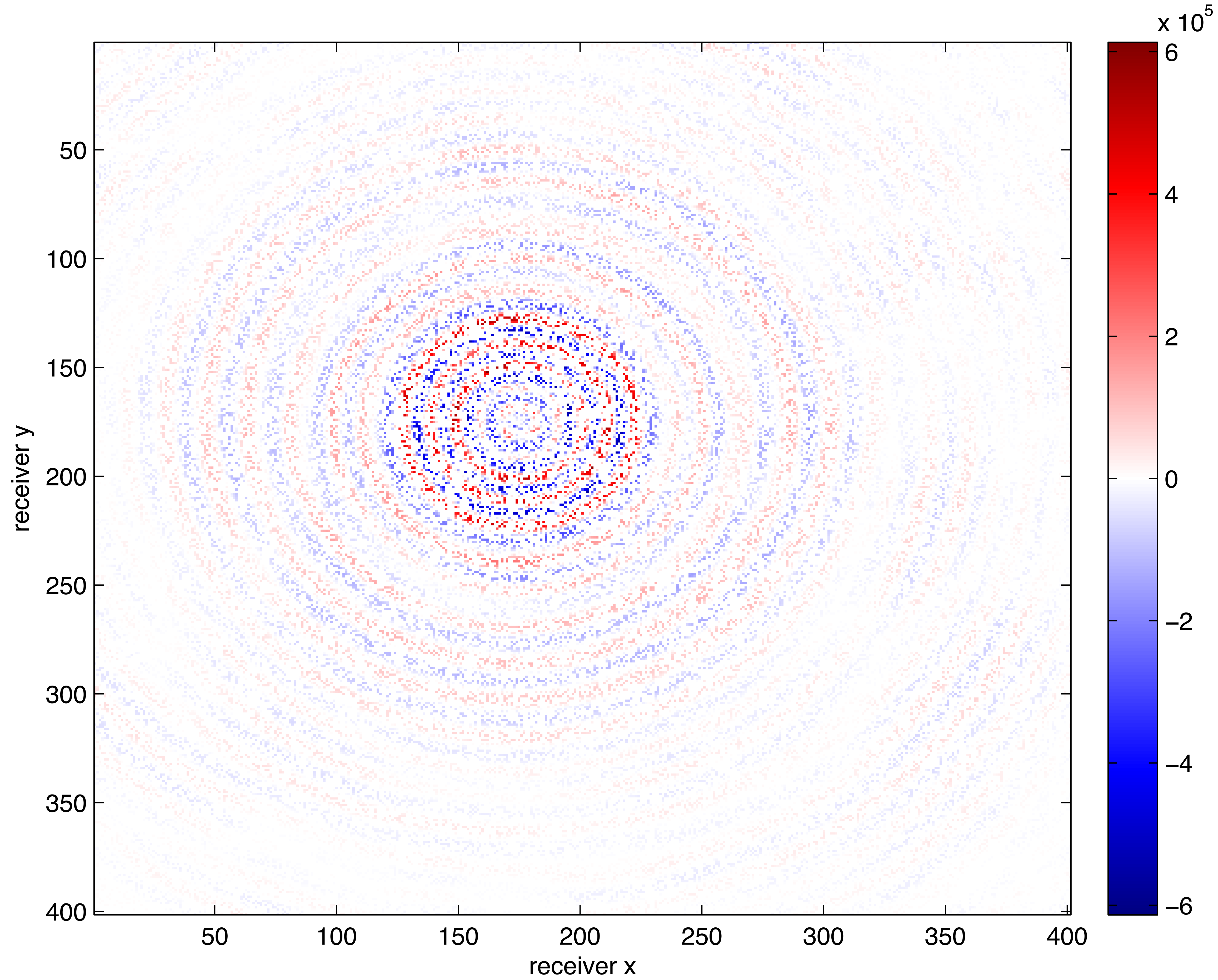
3 nodes, 4 processors per node

7.34Hz - 75% missing receivers

Common source gather



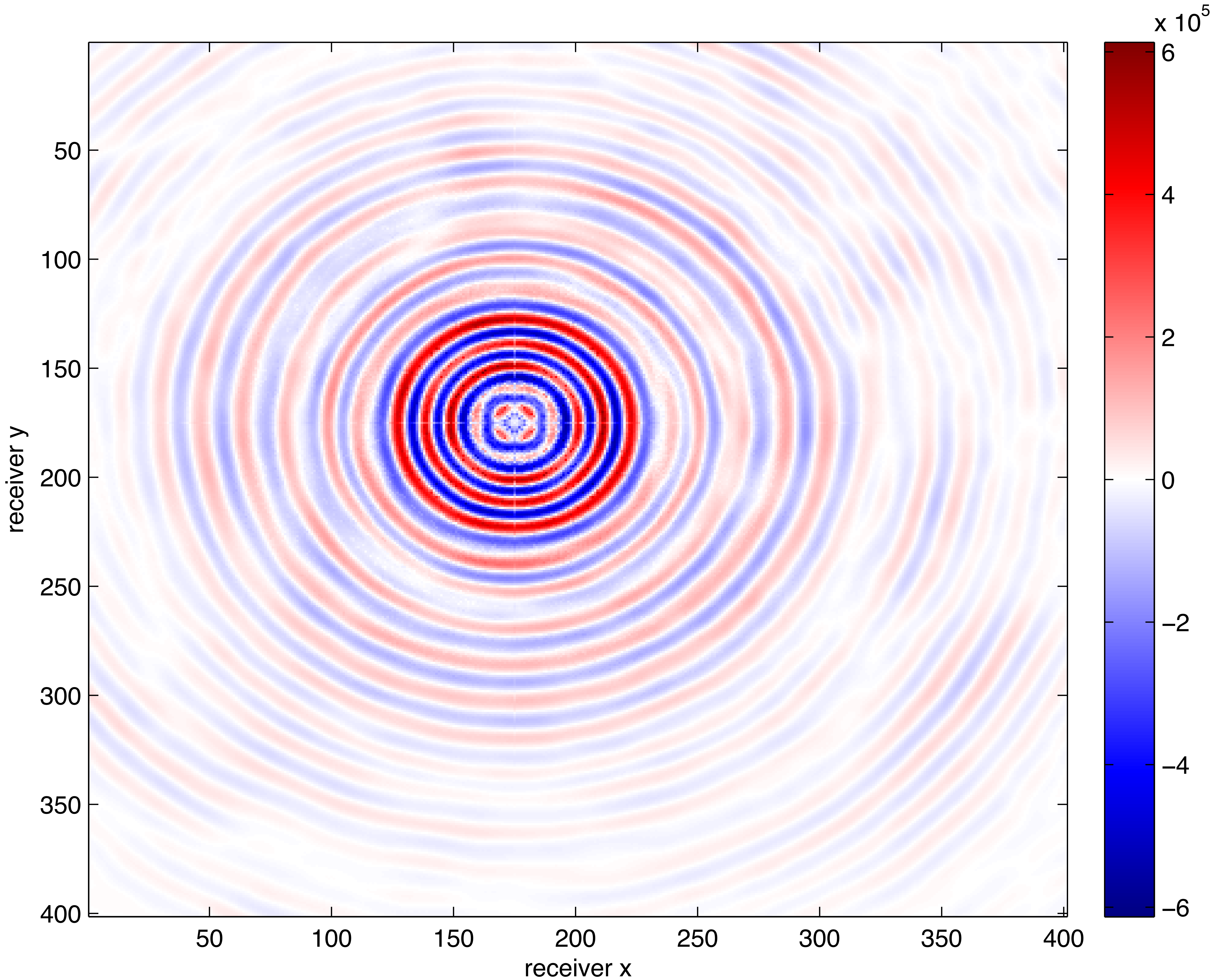
True data



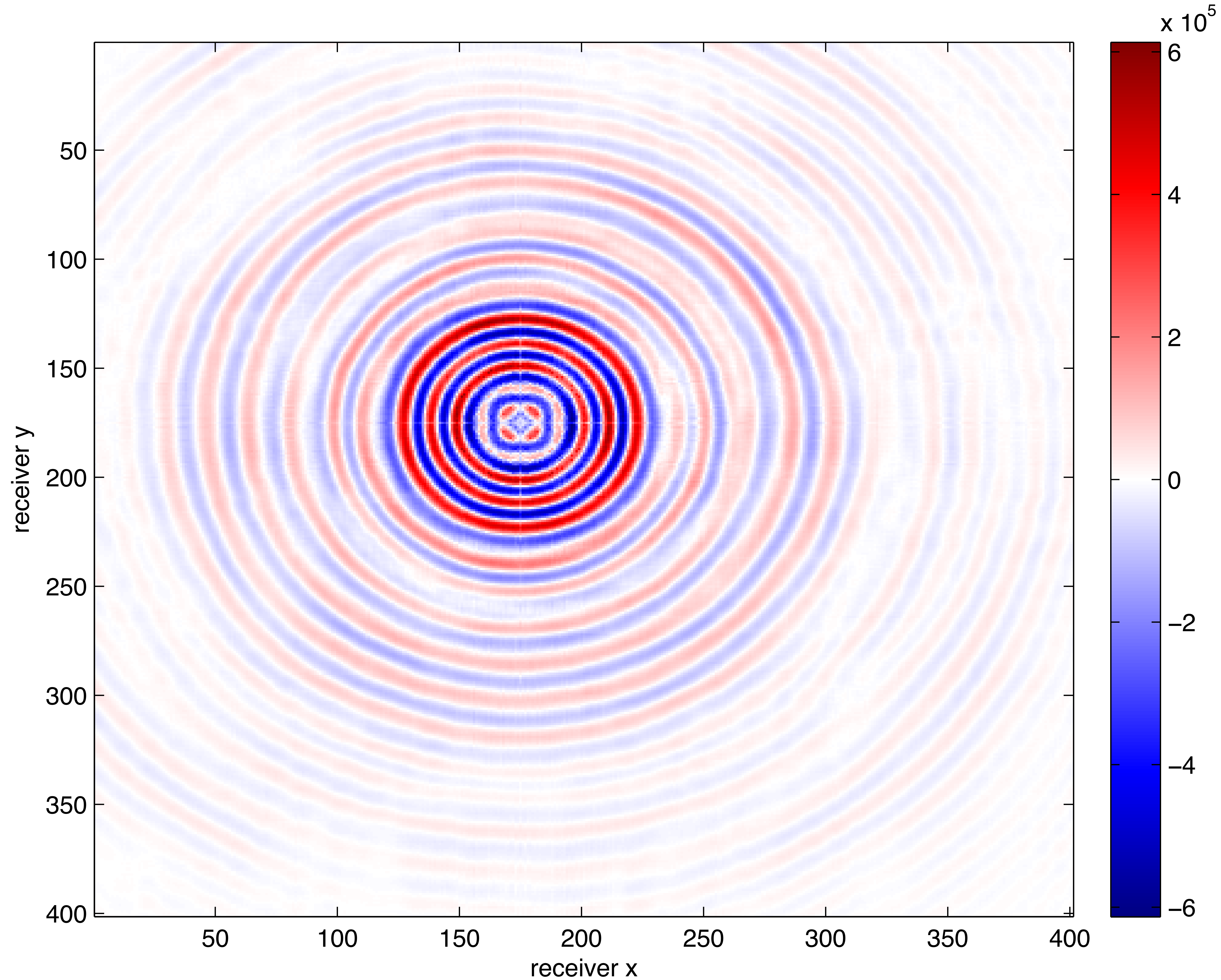
Subsampled data

7.34Hz - 75% missing receivers

Common source gather



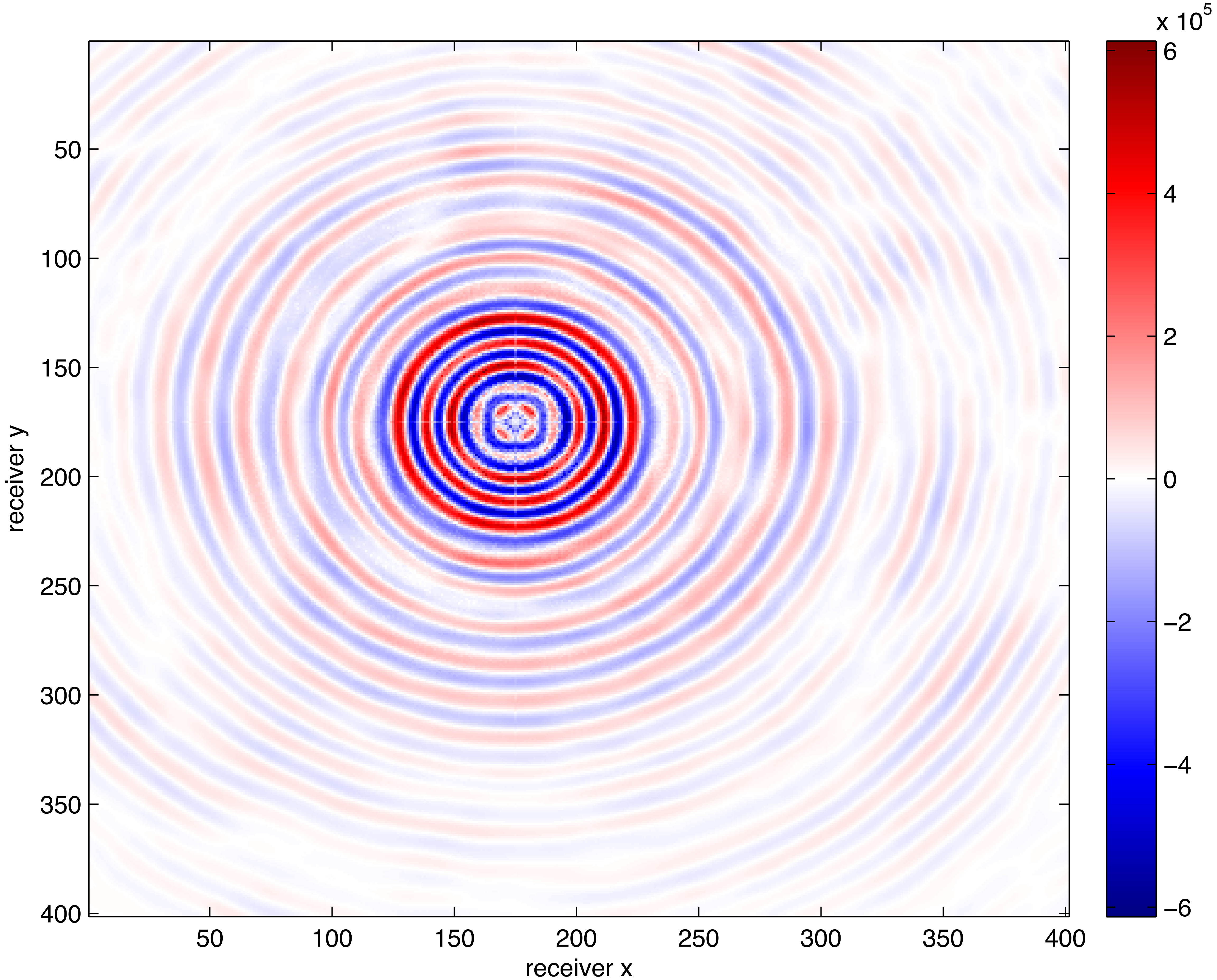
True data



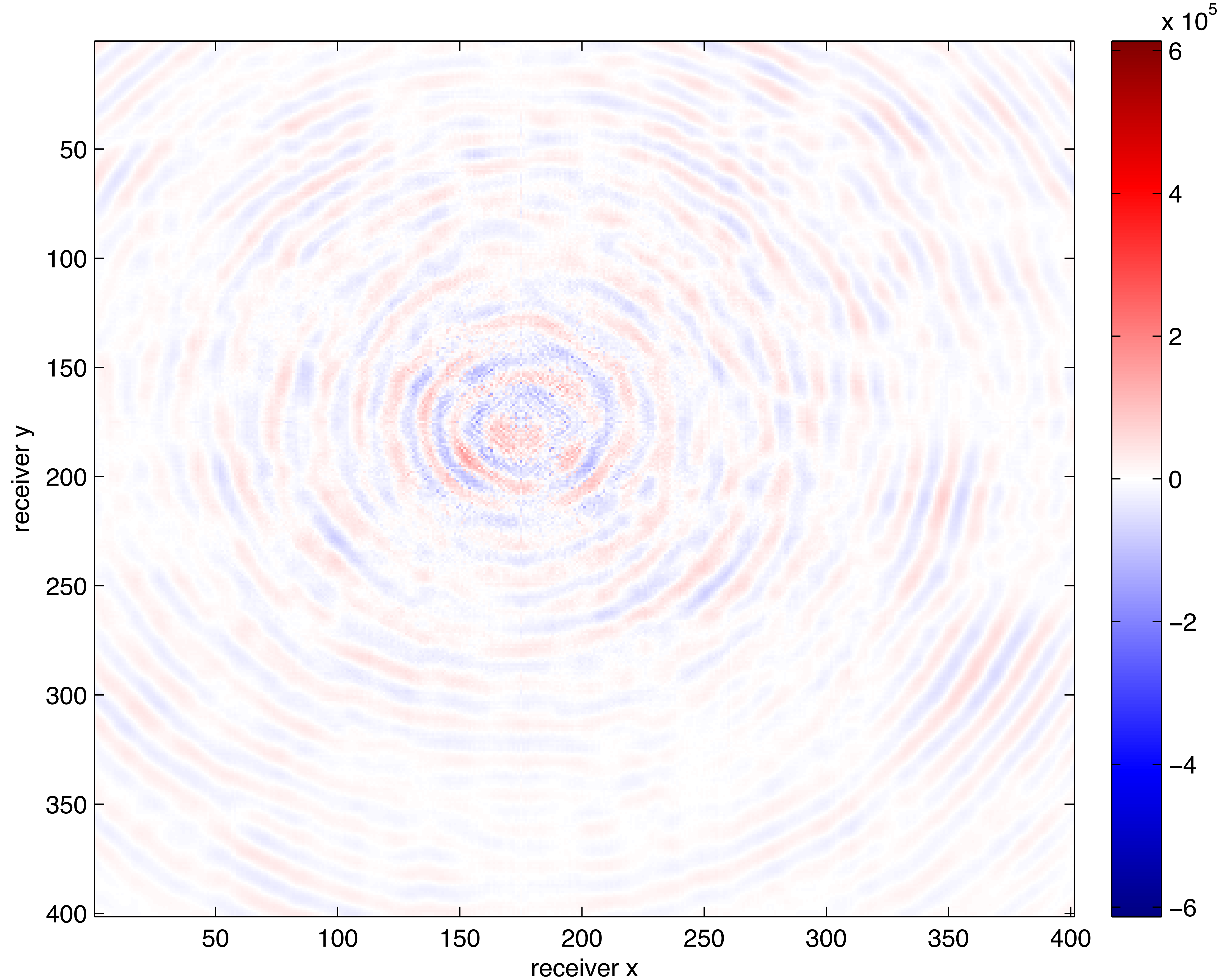
Interpolated data - SNR 14.5 dB

7.34Hz - 75% missing receivers

Common source gather



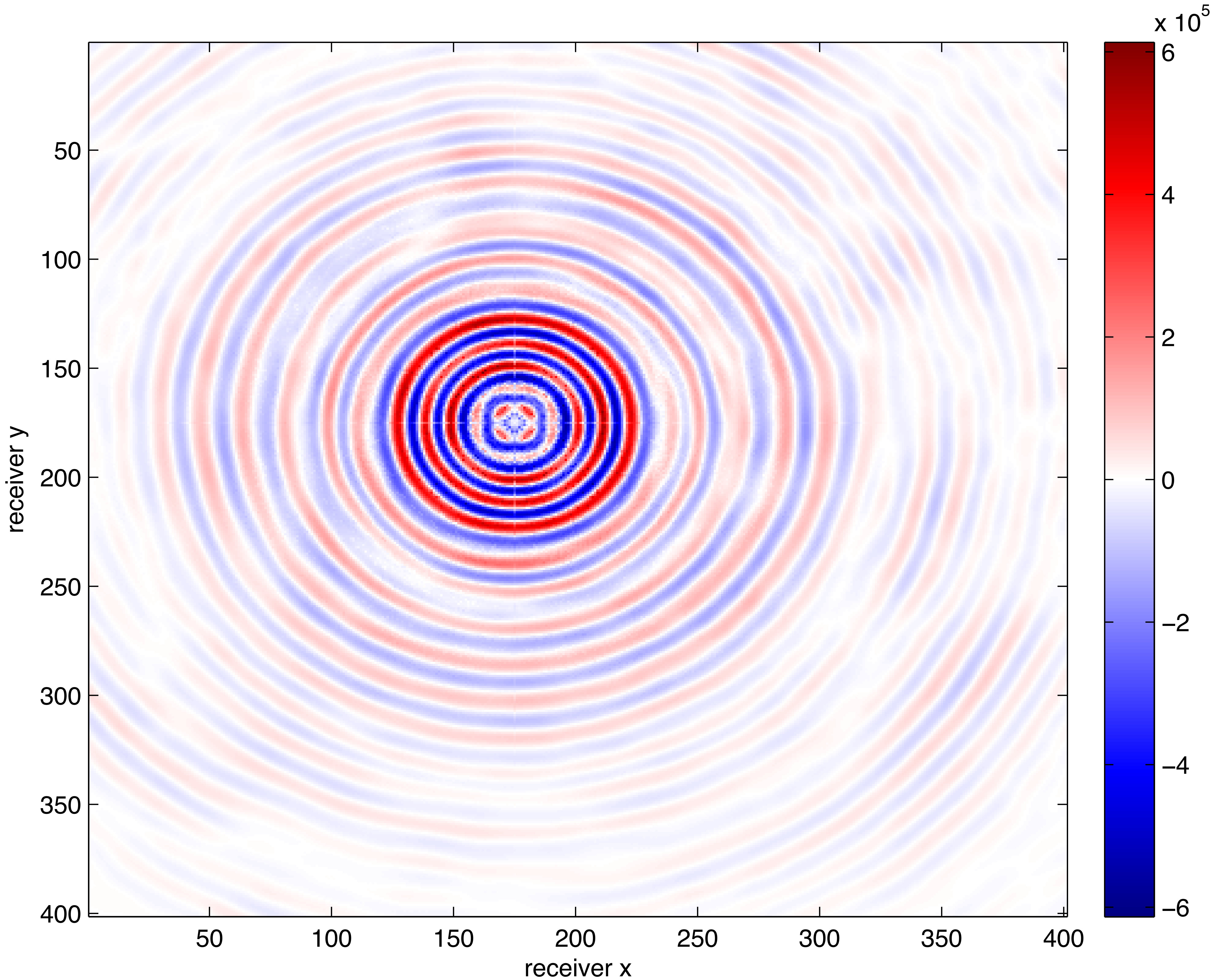
True data



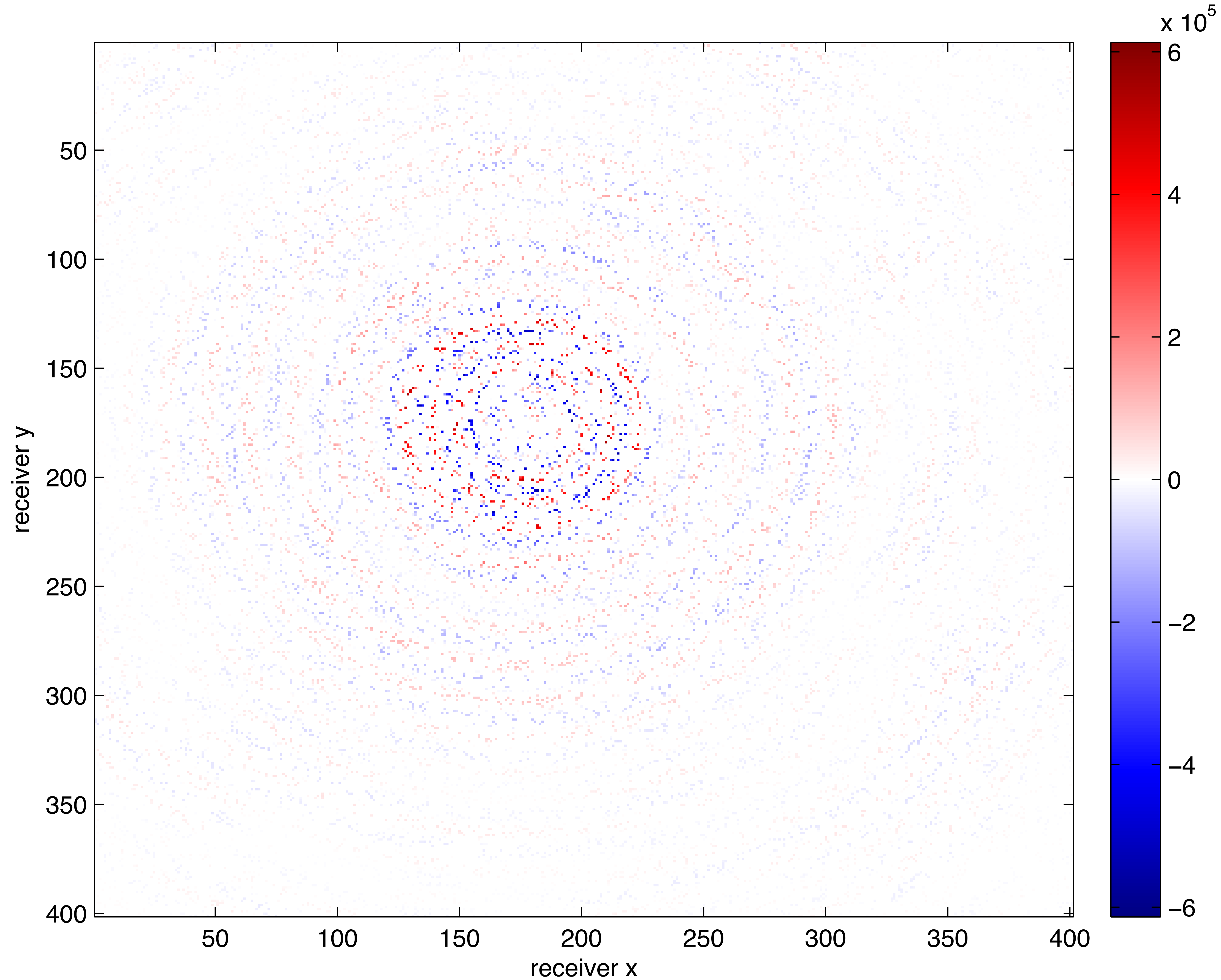
Difference

7.34Hz - 90% missing receivers

Common source gather



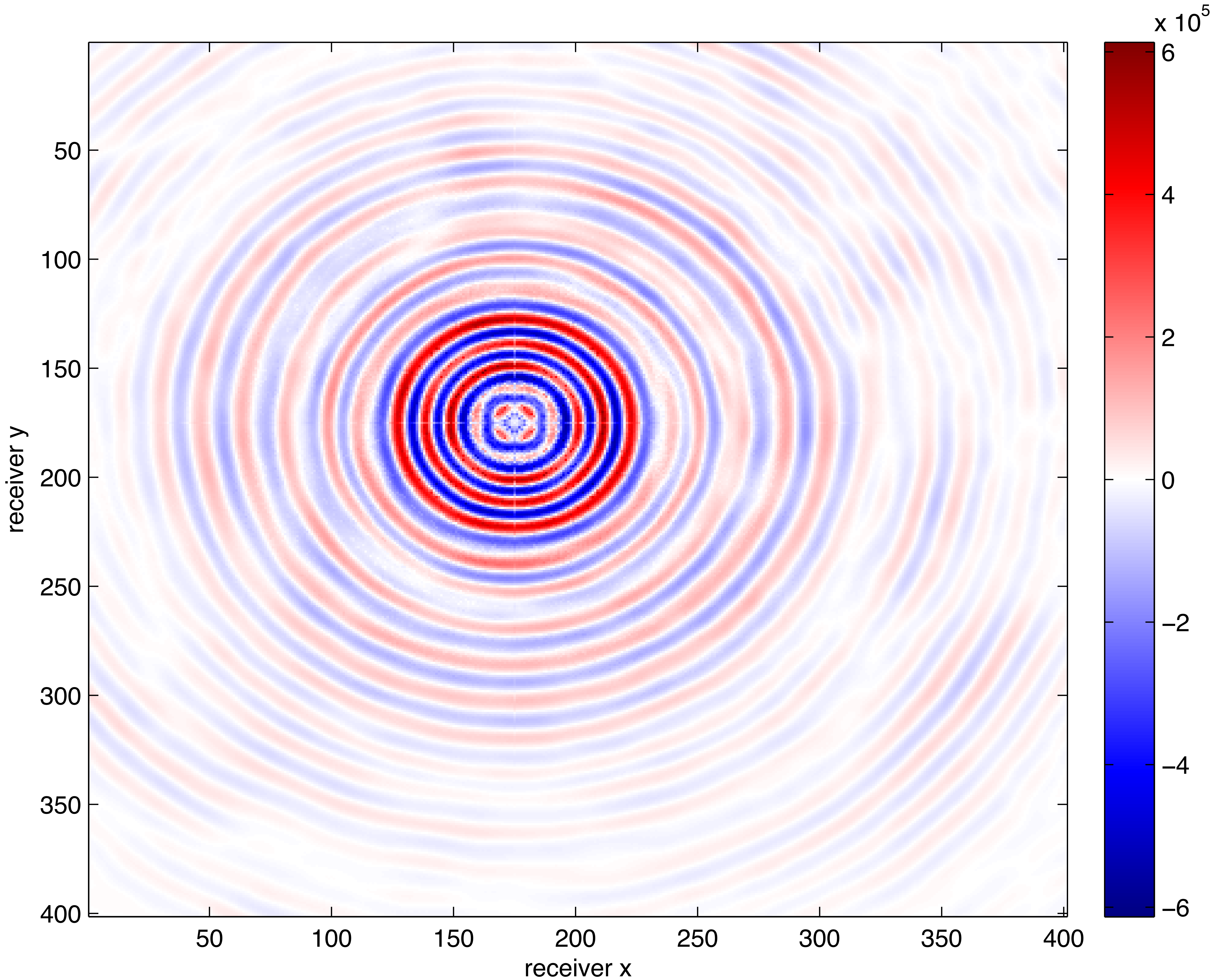
True data



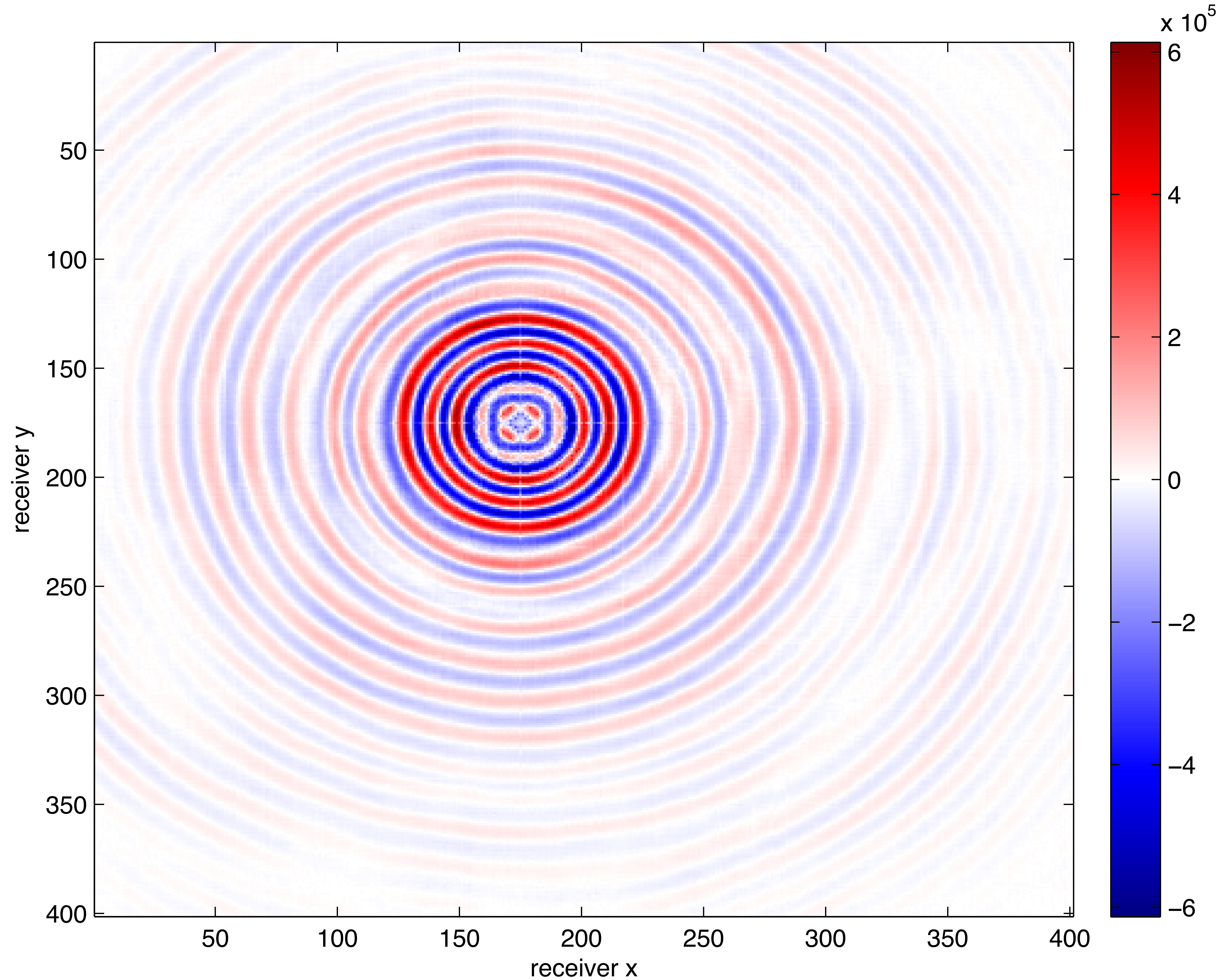
Subsampled data

7.34Hz - 90% missing receivers

Common source gather



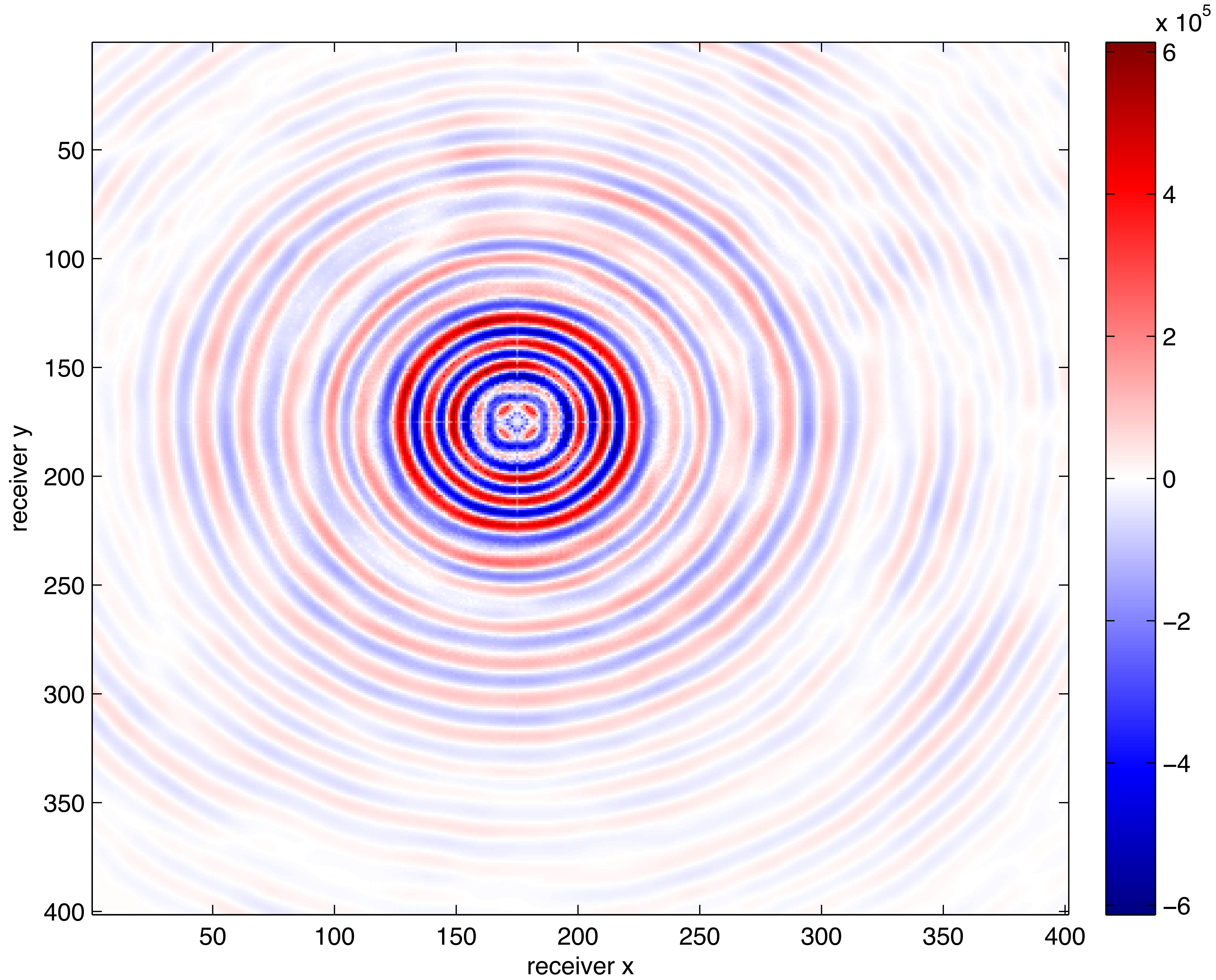
True data



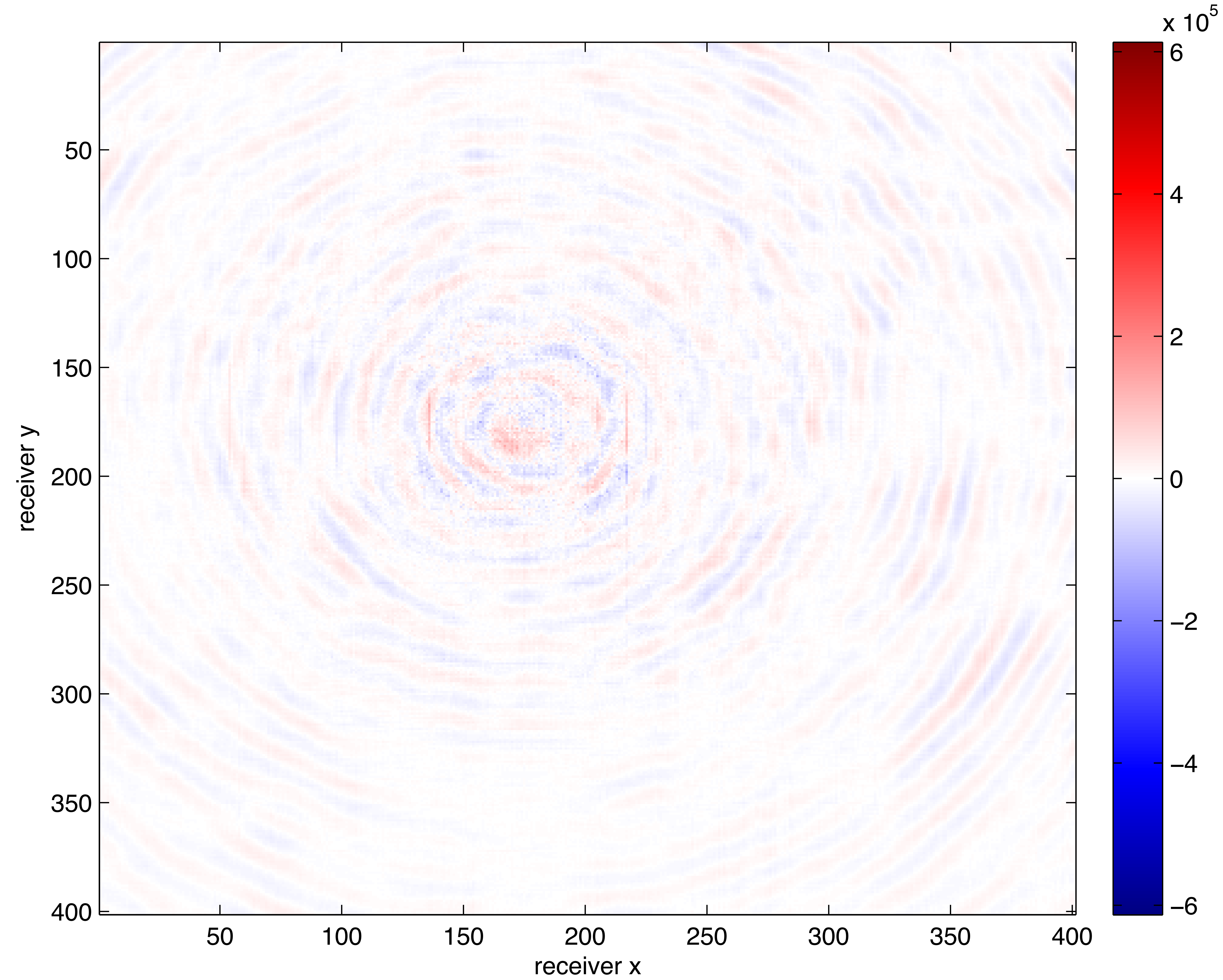
Interpolated data - SNR 15.9 dB

7.34Hz - 90% missing receivers

Common source gather



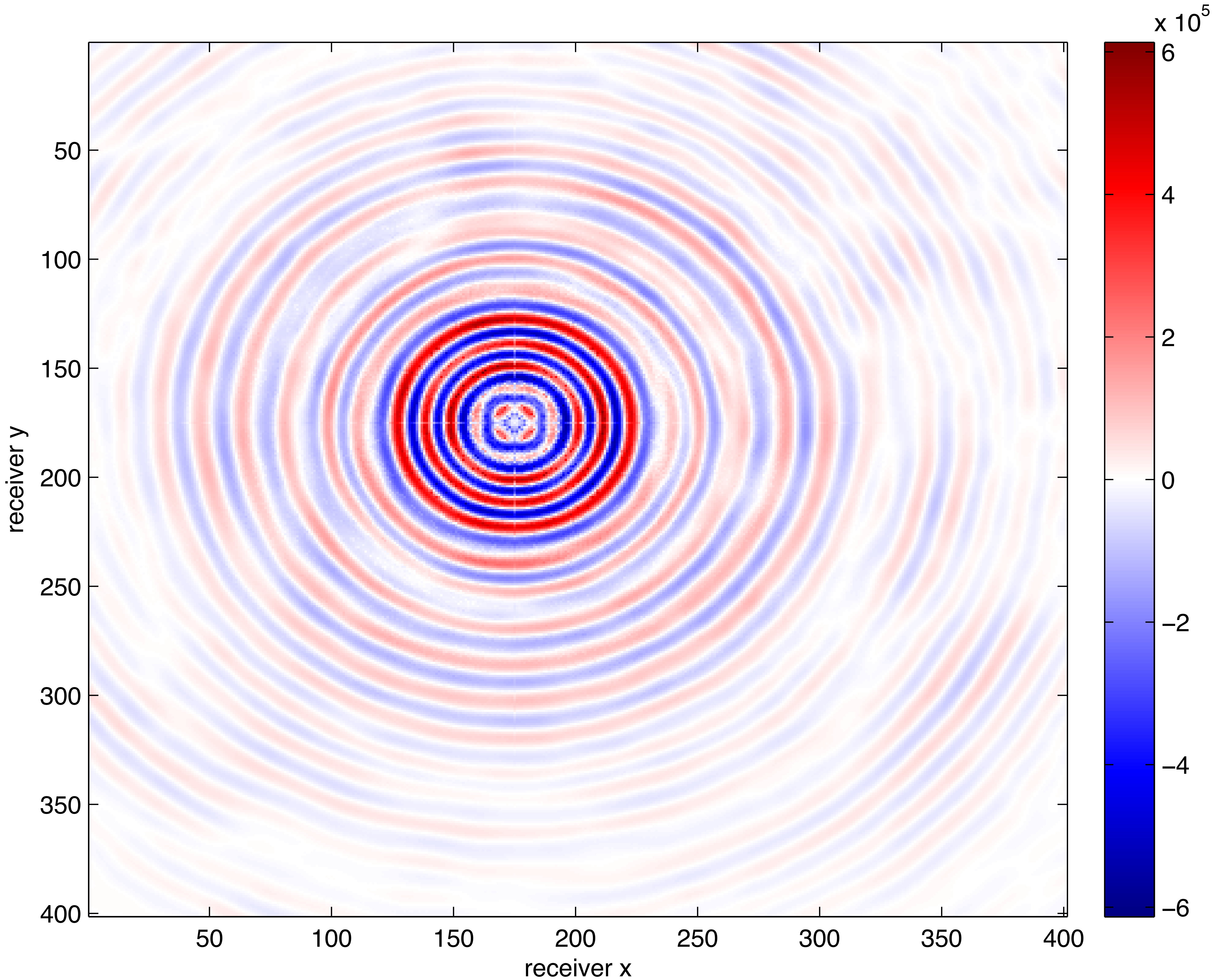
True data



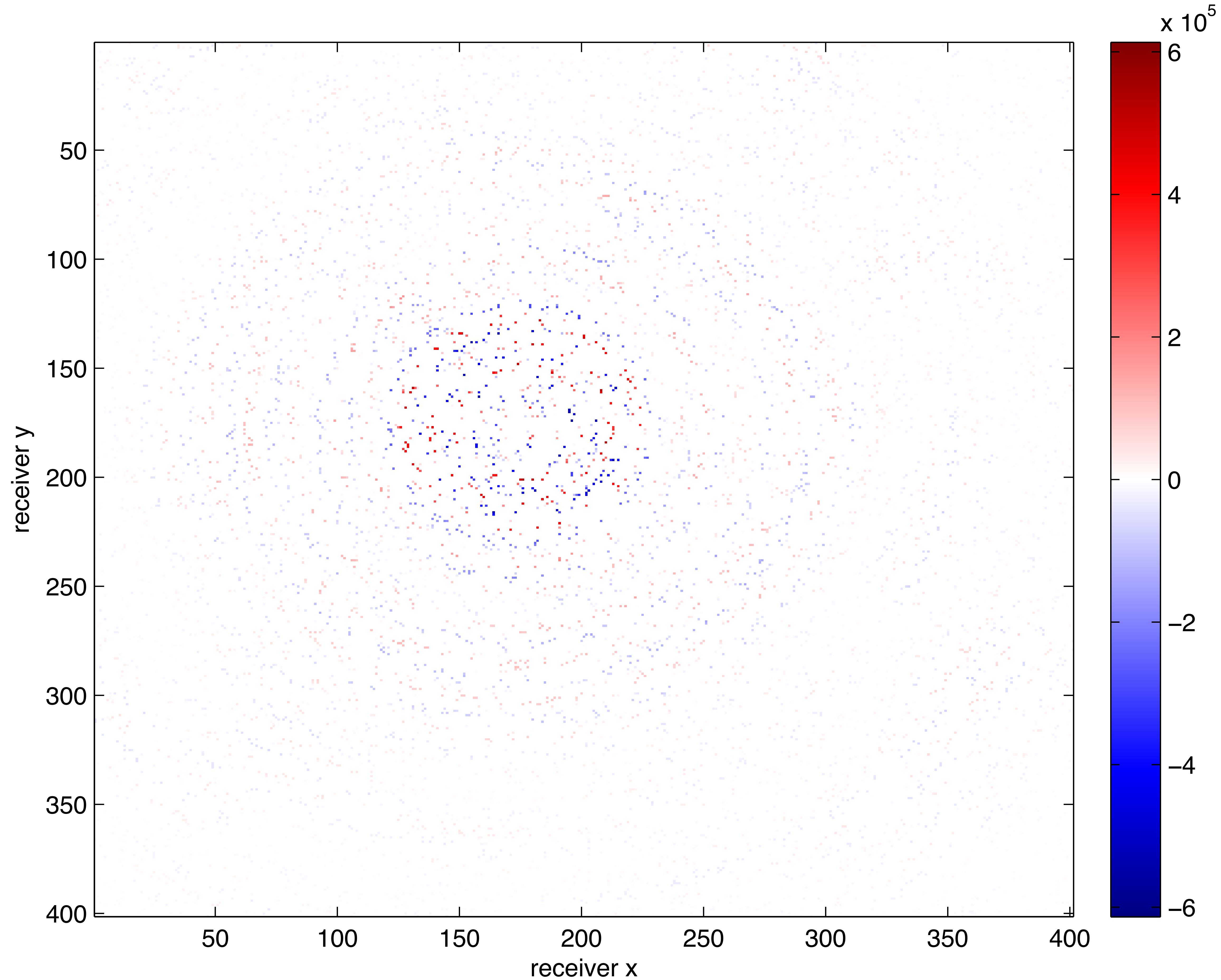
Difference

7.34Hz - 95% missing receivers

Common source gather



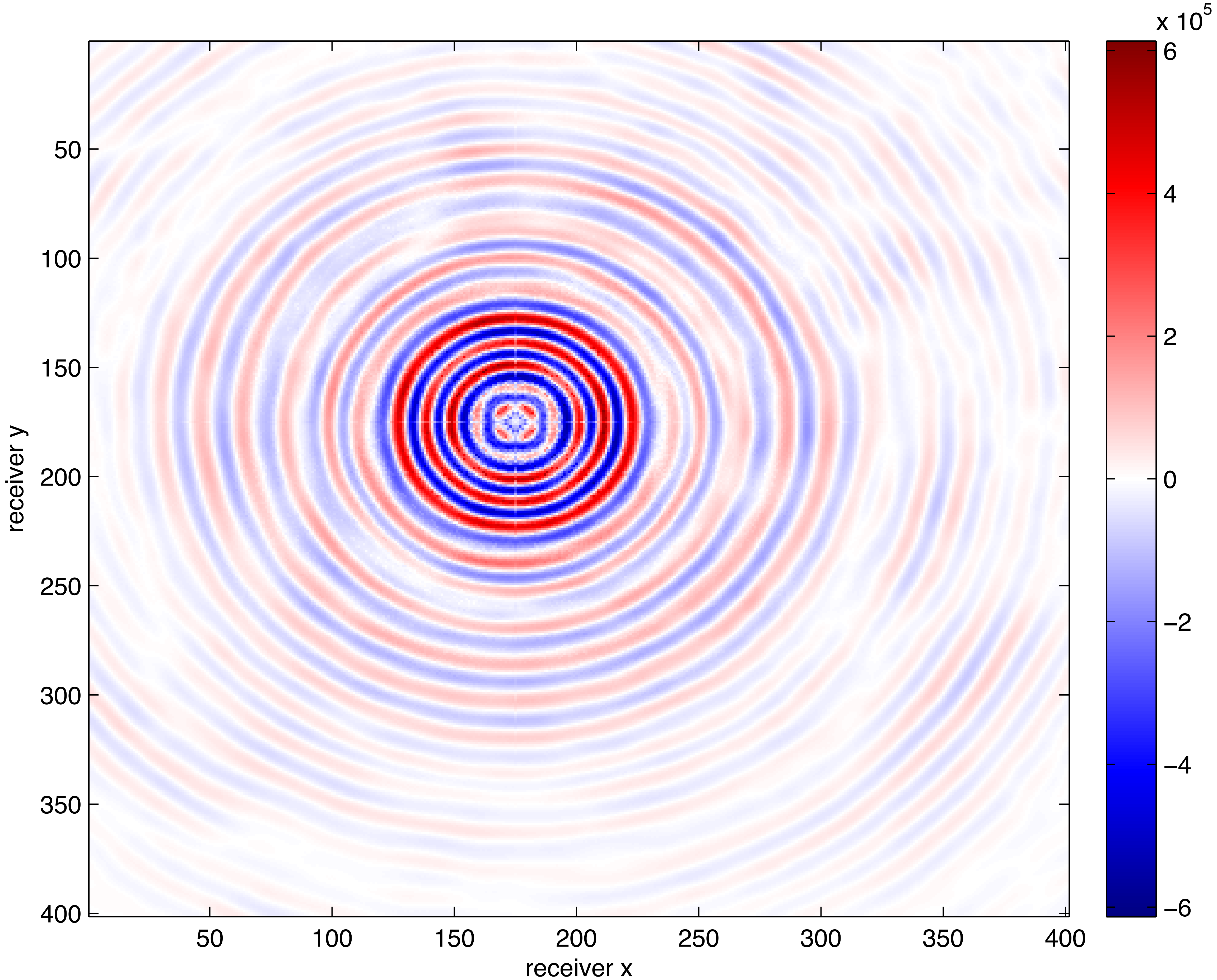
True data



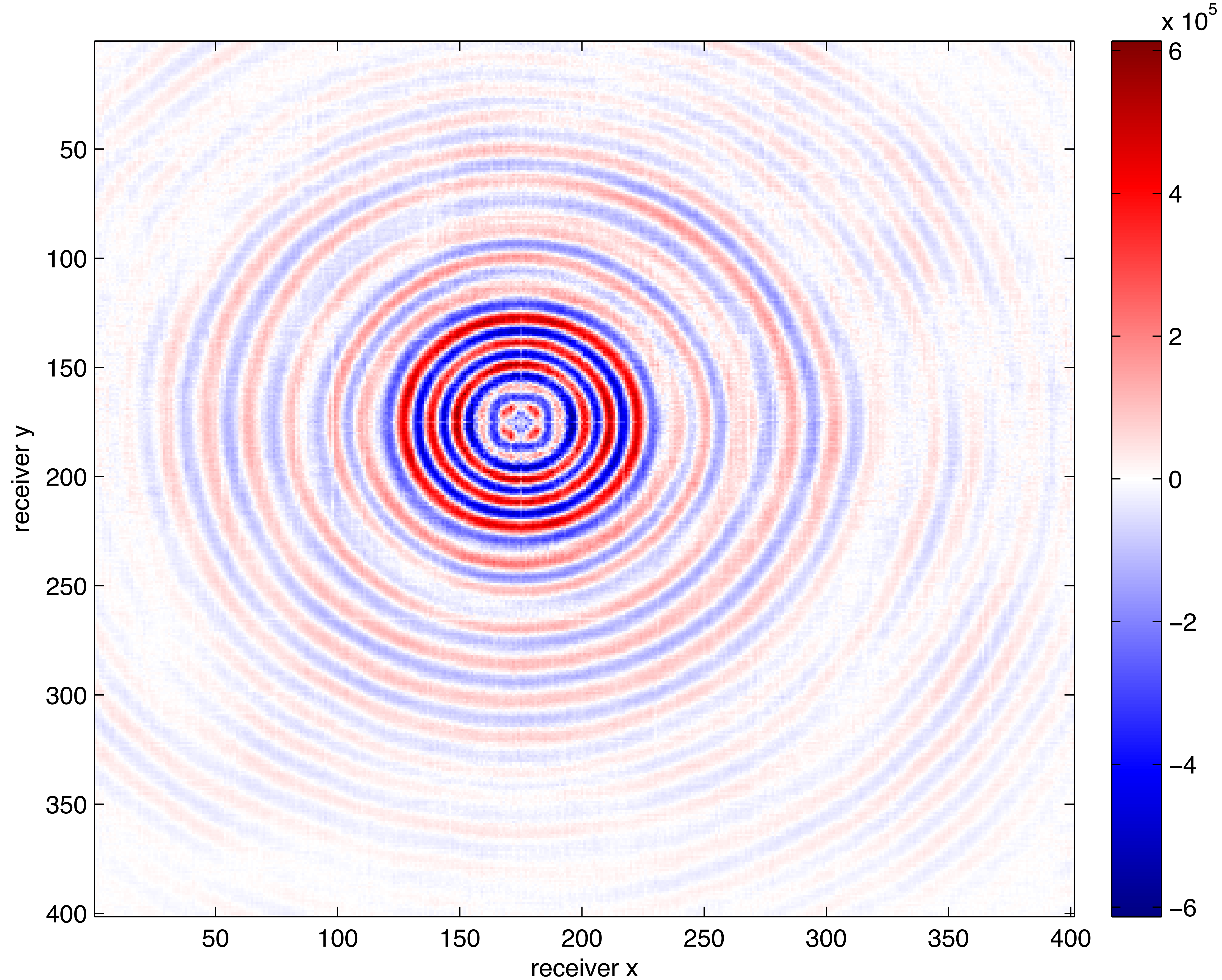
Subsampled data

7.34Hz - 95% missing receivers

Common source gather



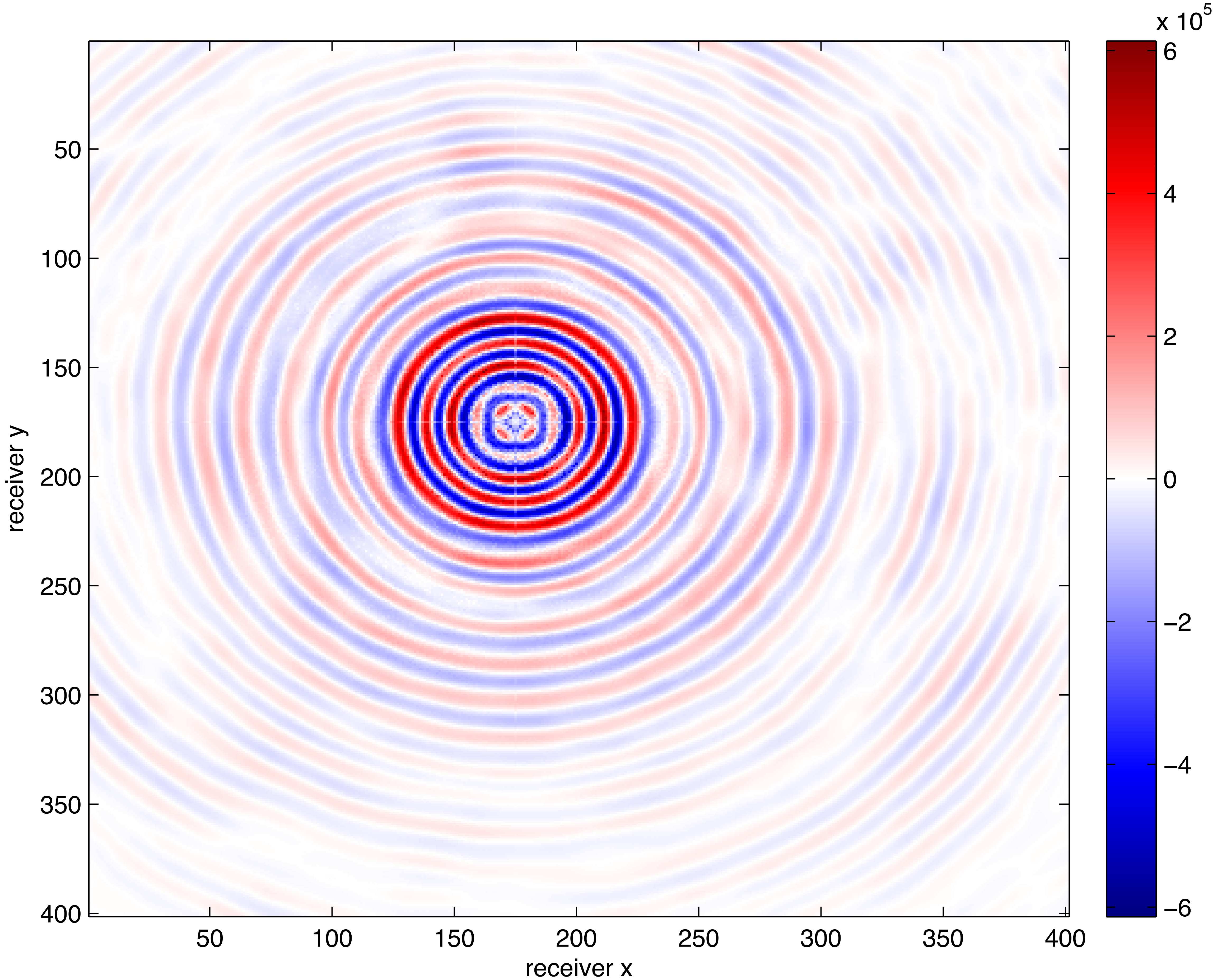
True data



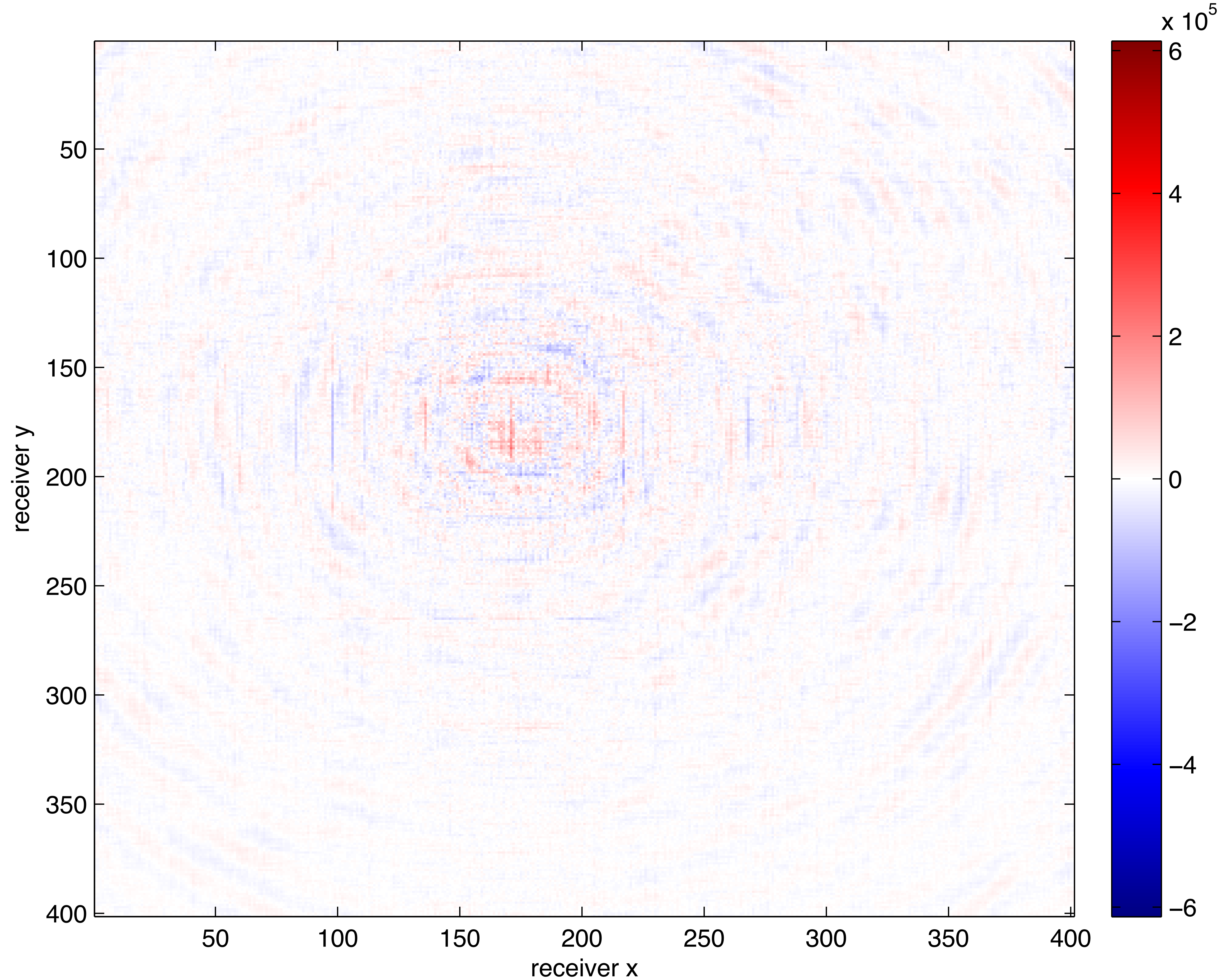
Interpolated data - SNR 14.2 dB

7.34Hz - 95% missing receivers

Common source gather



True data



Difference

BG data - 7.34Hz frequency slice

	SNR	Time (hr)	Rank
75% missing receivers	14.3	19.0	500
90% missing receivers	15.3	16.5	250
95% missing receivers	13.4	17.1	250

Straightforward extensions

Robust penalties for dealing with non-Gaussian noise

- huber, student's-t penalties, etc.

For blocks with very few data points, can exploit sparsity

Alternating LR - Next presentation by Oscar Lopez after the break

Off the grid tensor interpolation

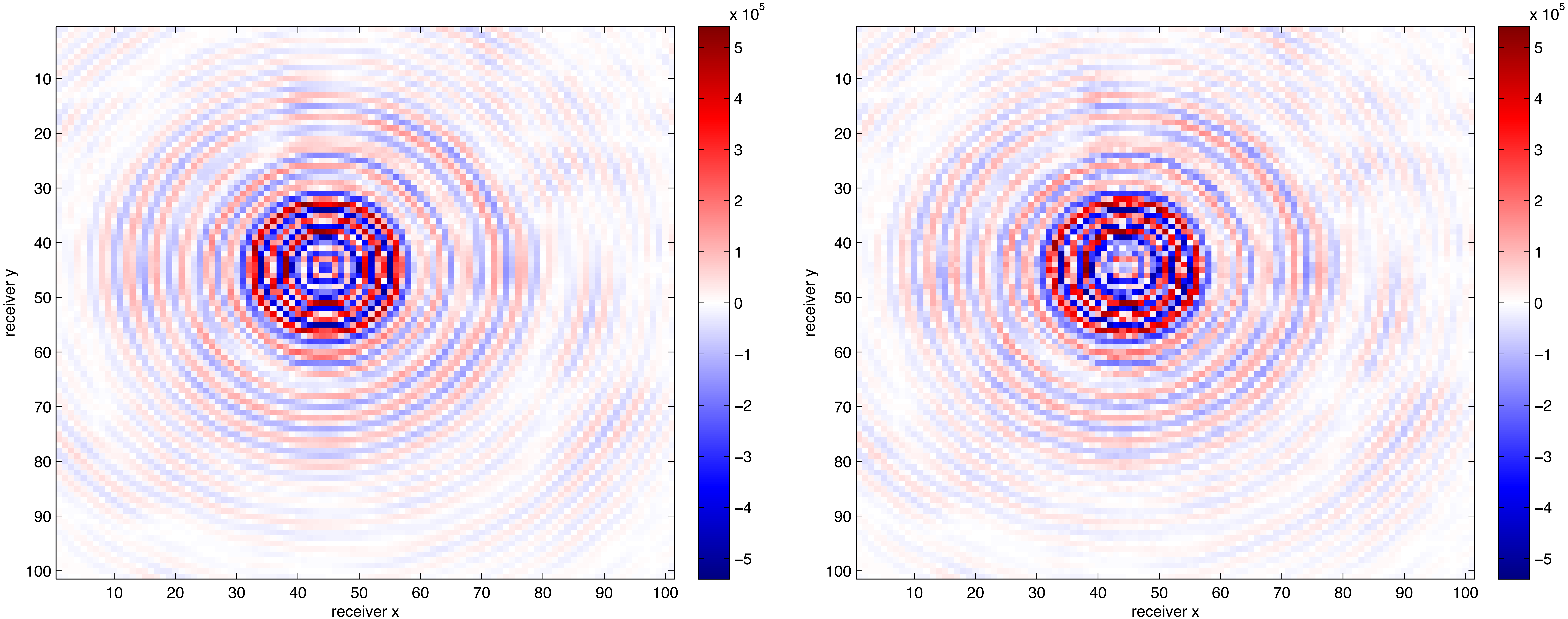
Regular vs irregular grid

Full data on a regular 401 x 401 m grid with 50m spacing

Subsampled data on an irregularly perturbed grid

- 200m spacing with 50% random 50m perturbations

Regular vs irregular grid
Common source gather



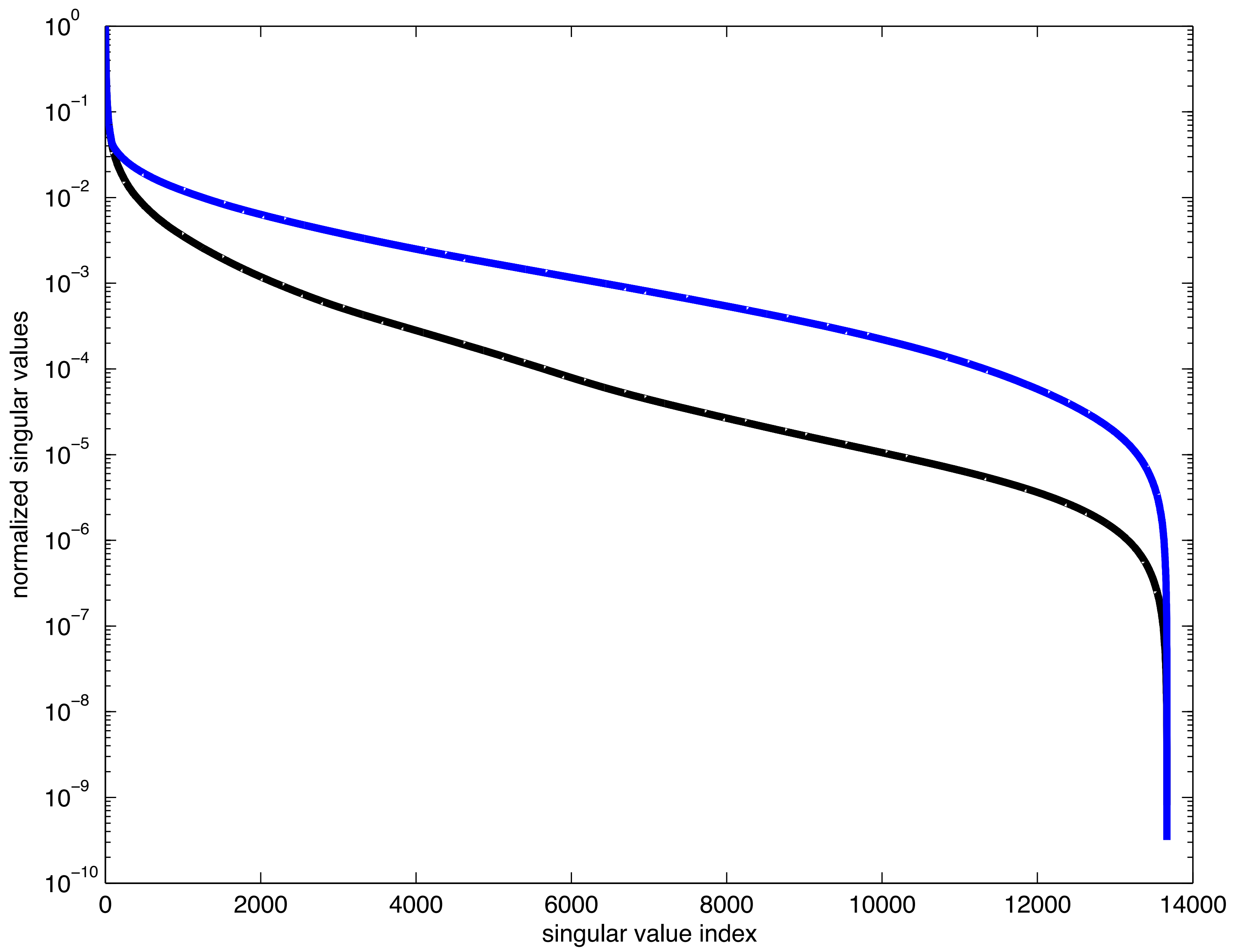
Regular grid

Irregular grid

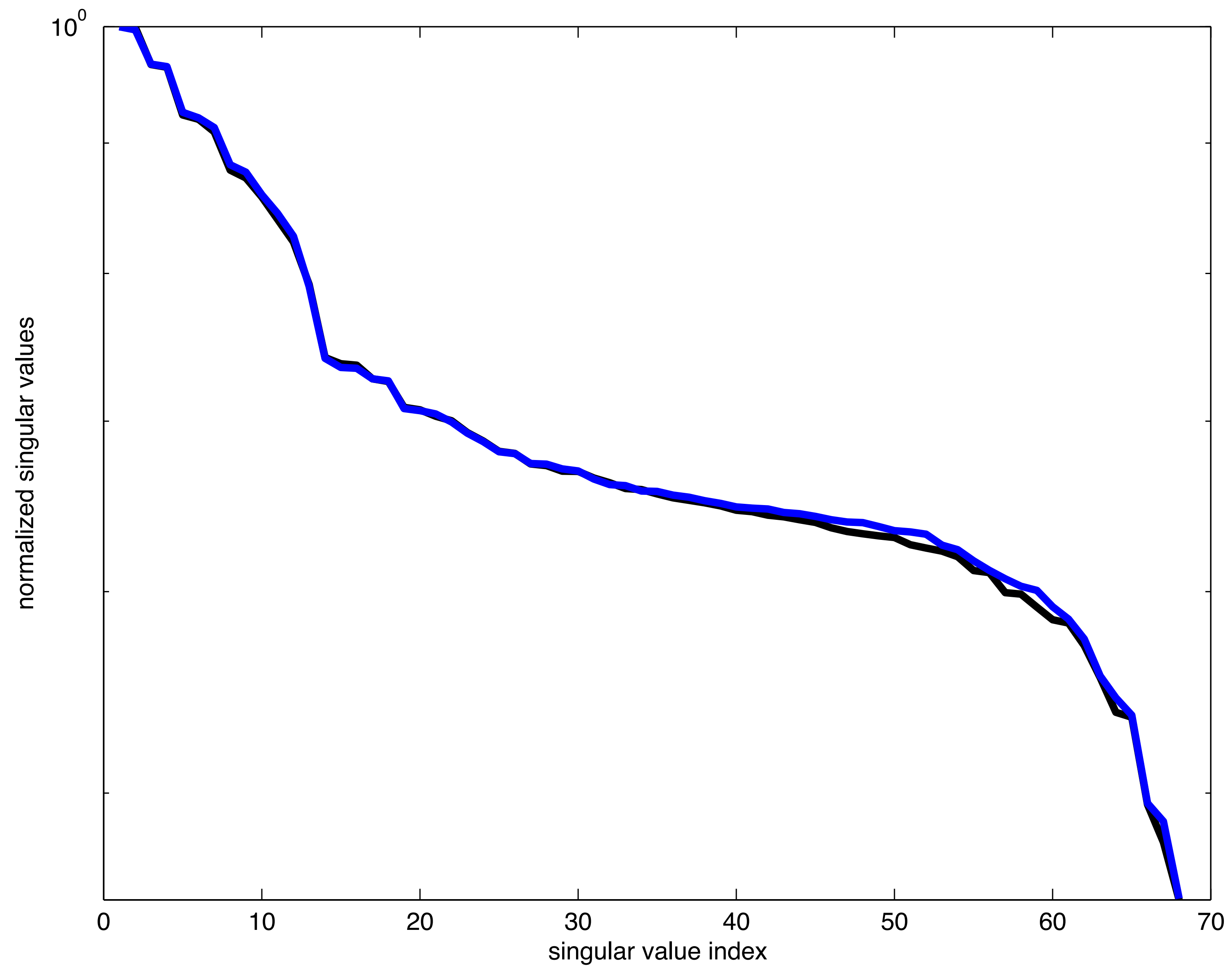
Regular vs irregular grid - singular values

Black - regular grid

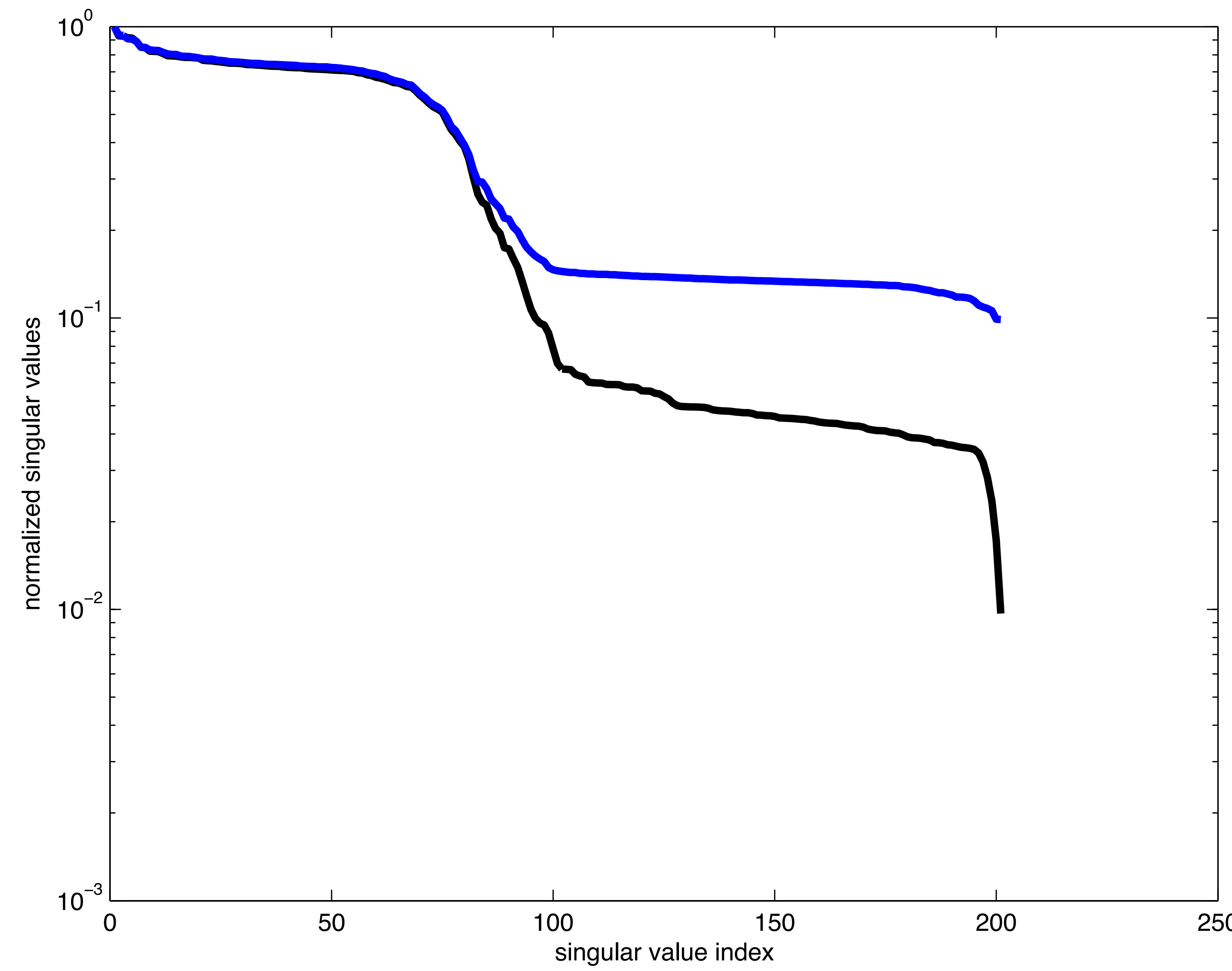
Blue - irregular grid



source x, receiver x



source x



receiver x

Off the grid tensor interpolation

The data volume is **no longer** low rank when irregularly sampled

- standard tensor completion framework won't work well

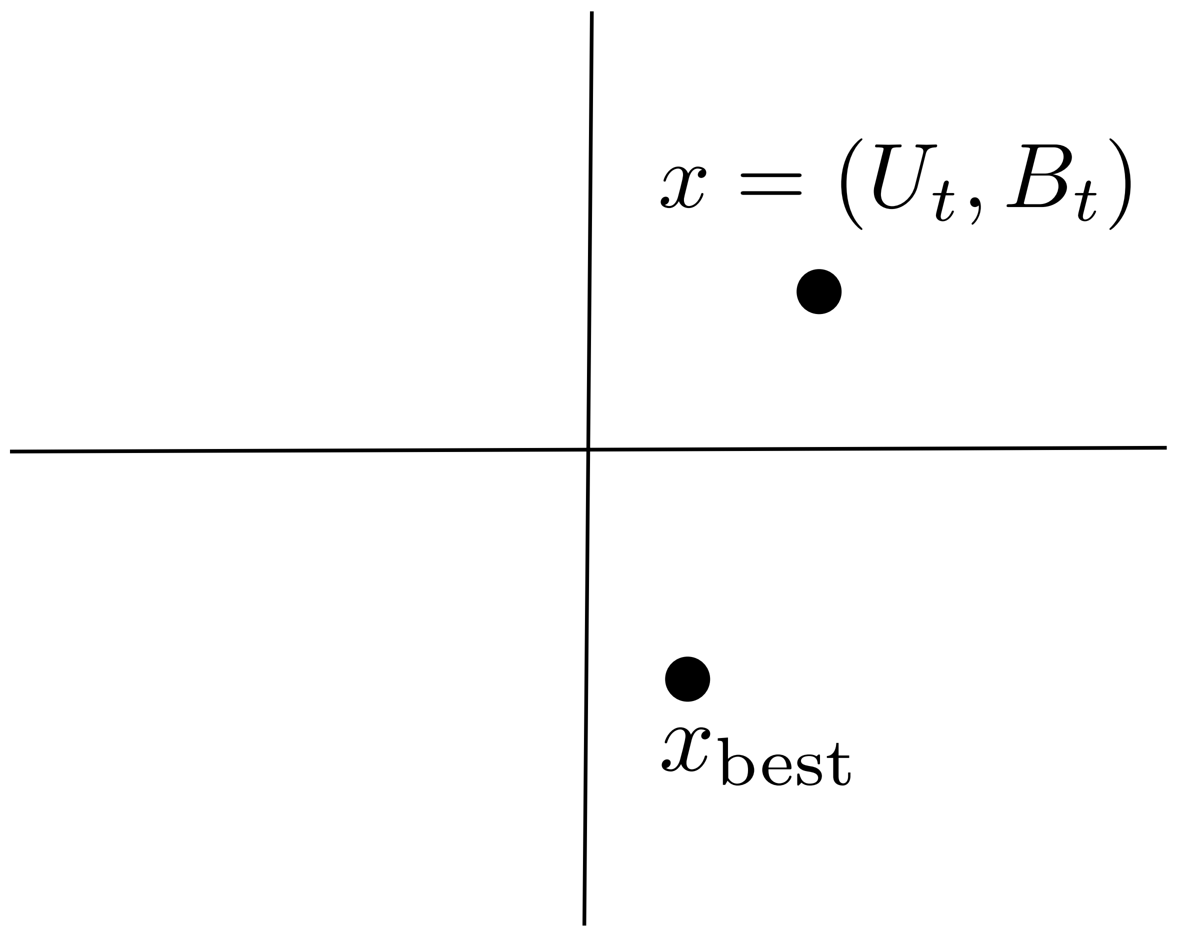
Solution

- construct a domain where the data **is** low rank
- choose an appropriate transform : low rank domain -> sampling domain
- incorporate transform in to optimization problem

Optimization program

$\mathbb{C}^D$

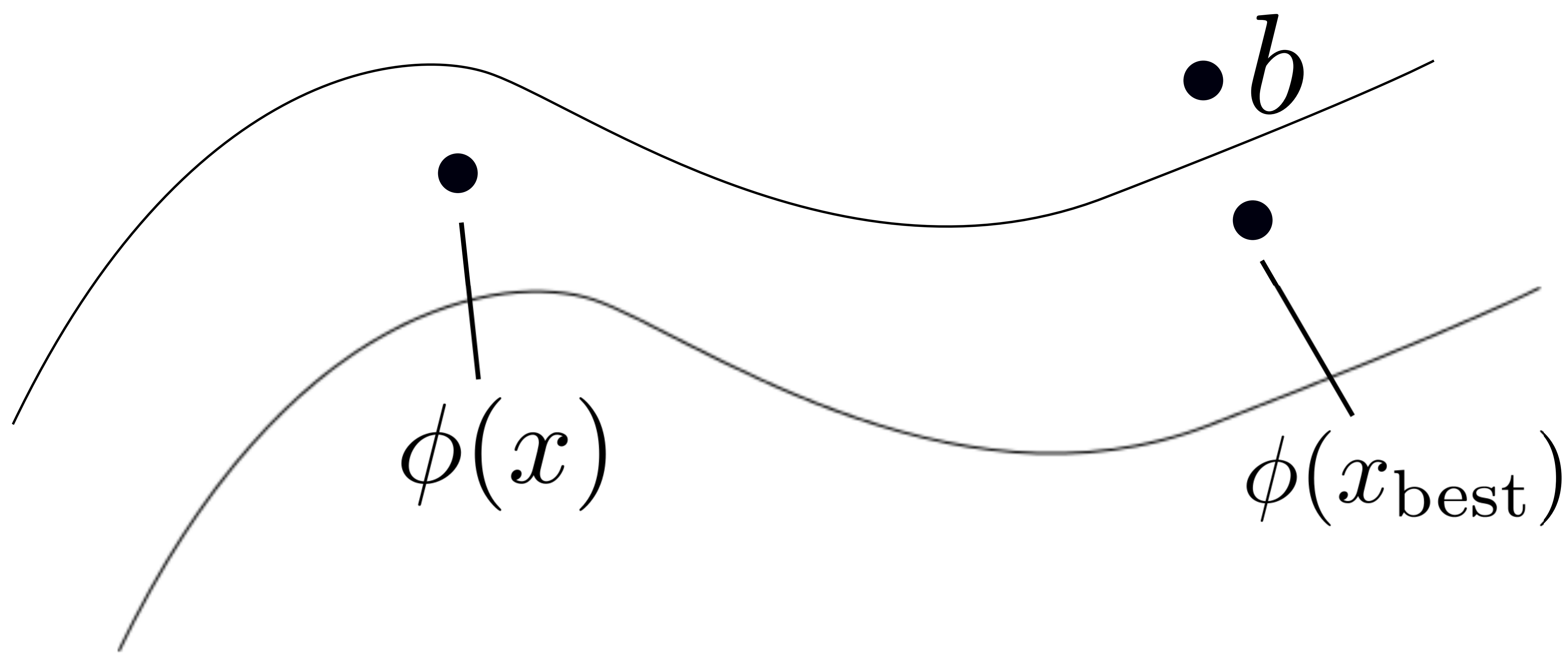
$$\phi(x)$$



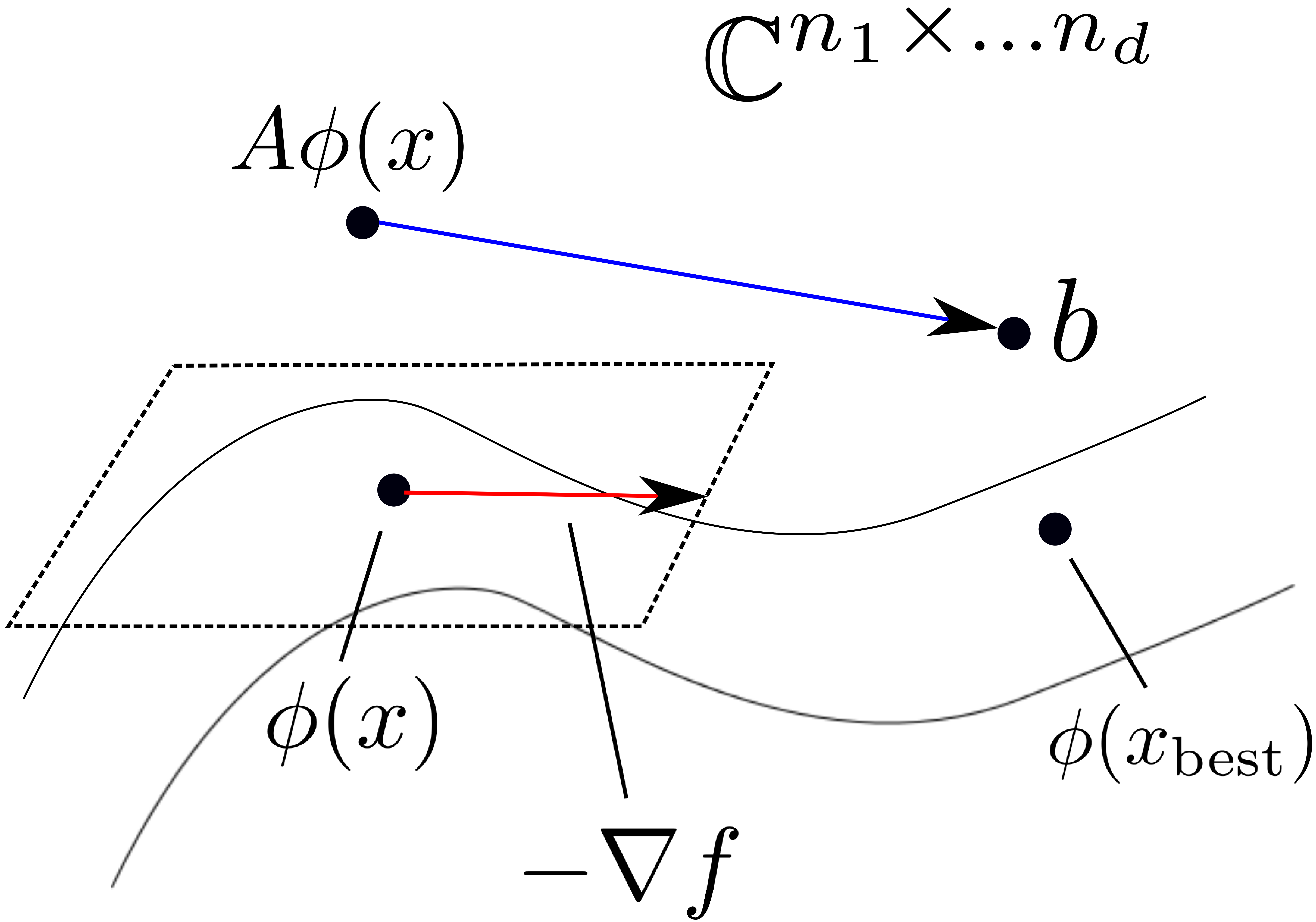
Full-tensor space

$$\mathbb{C}^{n_1 \times \dots n_d}$$

Parameter space



Optimization program



Optimization problem

The standard problem we solve is

$$\min_x \|\mathcal{A}\phi(x) - b\|_2^2$$

Our sampling operator is typically

$$\mathcal{A} = \mathcal{R}\mathcal{P}$$

where

$$\mathcal{R} : \text{regular full grid} \rightarrow \text{subsampled grid}$$

$$\mathcal{P} : (\text{src x}, \text{rec x}, \text{src y}, \text{rec y}) \rightarrow (\text{src x}, \text{src y}, \text{rec x}, \text{rec y})$$

Optimization problem

In the irregular grid case, the subsampling operator is in fact

$$\mathcal{R} : \text{irregular full grid} \rightarrow \text{subsampled grid}$$

In order to take this discrepancy in to account, we introduce an operator

$$\mathcal{F} : \text{regular full grid} \rightarrow \text{irregular full grid}$$

Optimization problem

The sequence of operators is then

$$\mathcal{R} : \text{irregular full grid} \rightarrow \text{subsampled grid}$$
$$\mathcal{F} : \text{regular full grid} \rightarrow \text{irregular full grid}$$
$$\mathcal{P} : (\text{src x}, \text{rec x}, \text{src y}, \text{rec y}) \rightarrow (\text{src x}, \text{src y}, \text{rec x}, \text{rec y})$$

We set $\mathcal{A} = \mathcal{RFP}$ and use the same optimization code as previously

In our examples, we use the non-uniform Fourier transform

Results

SLIM

Experiment setup

BG Group data

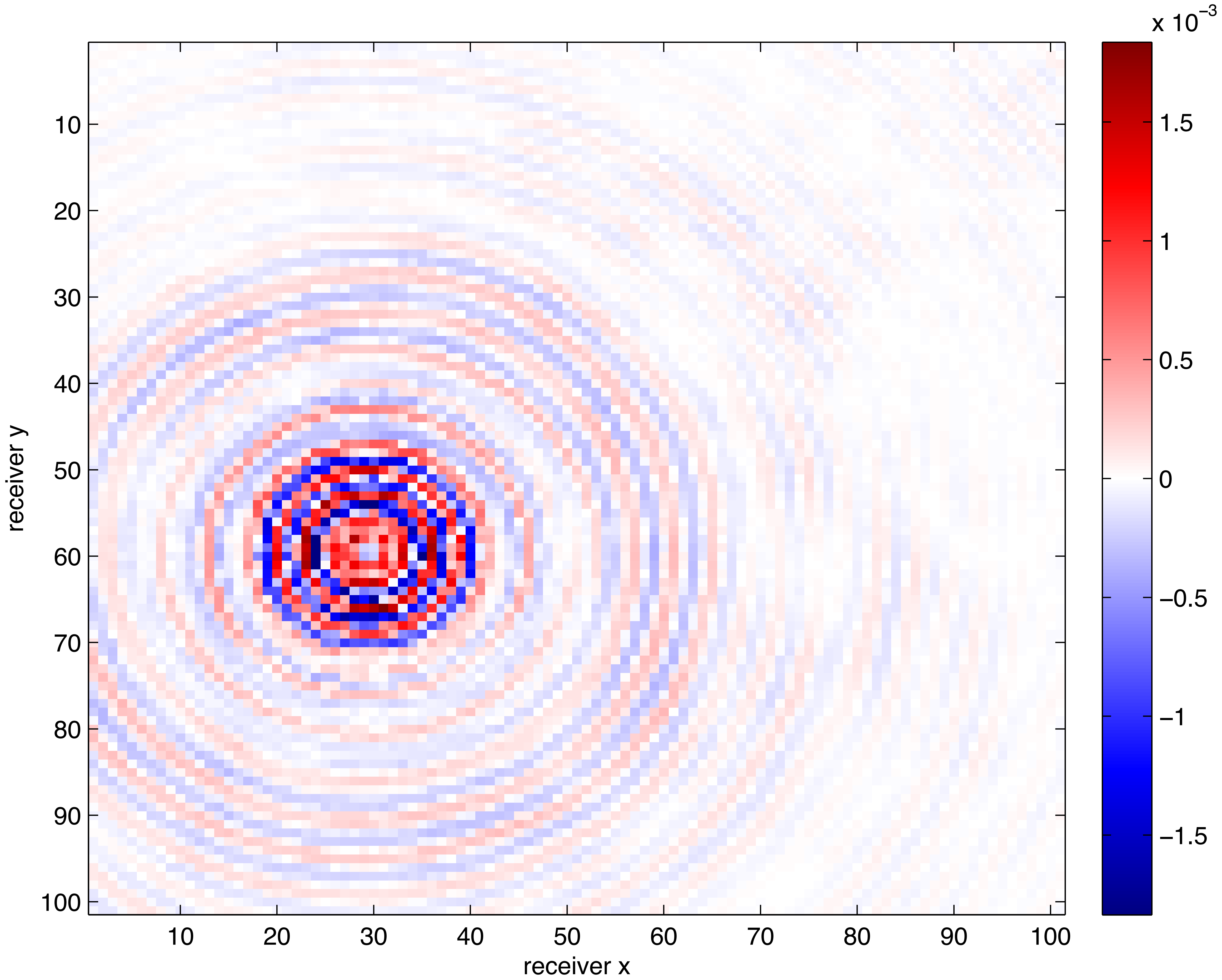
- 68 x 68 sources with 150m spacing
- 401 x 401 receivers with 50m spacing

Subsampled data on an irregularly perturbed grid

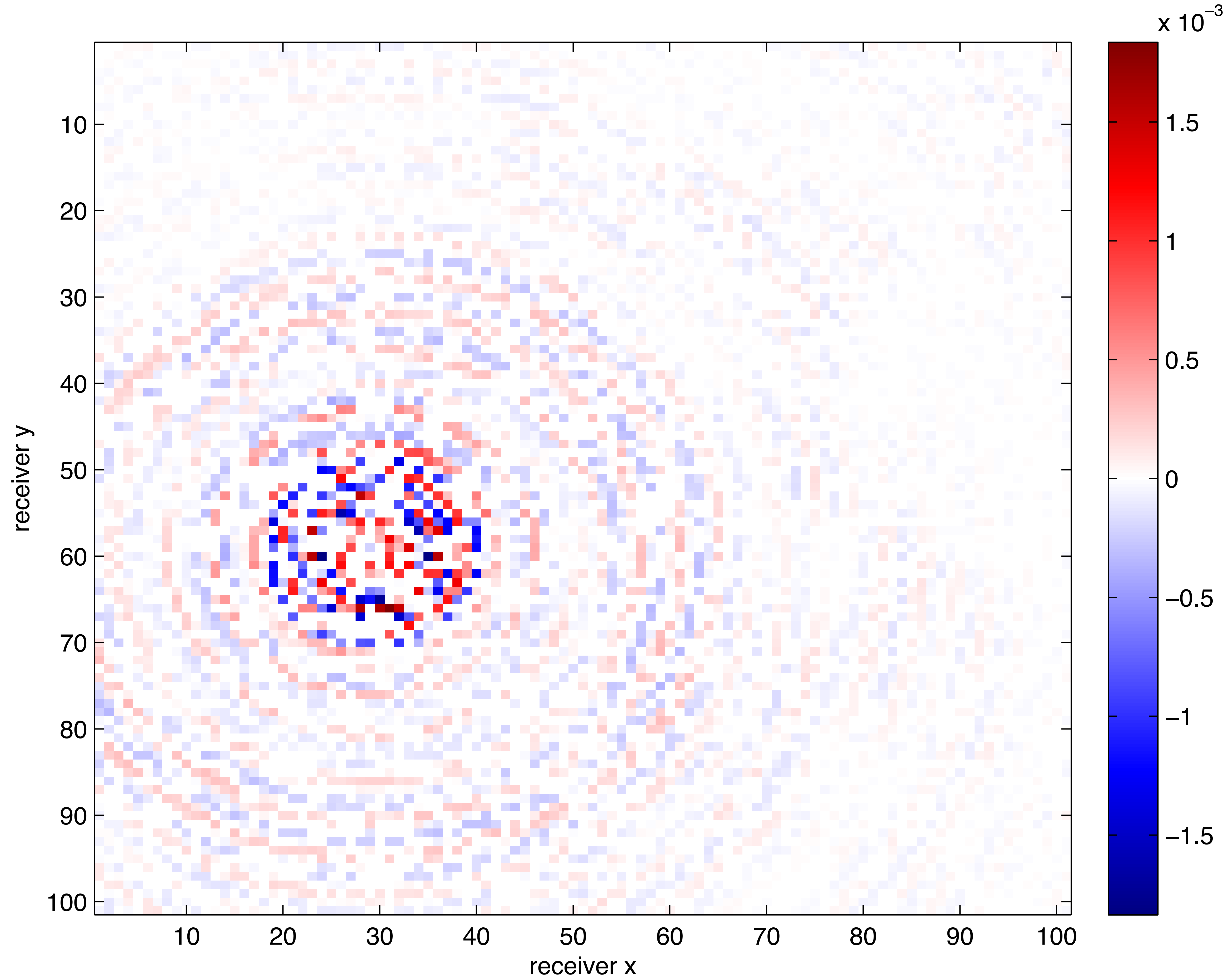
- source grid remains the same
- receiver grid subsampled to 200m spacing with 50% random 50m perturbations

Removed 50% of receivers, recovered with HT optimization

Regularized recovery - 50% missing receivers

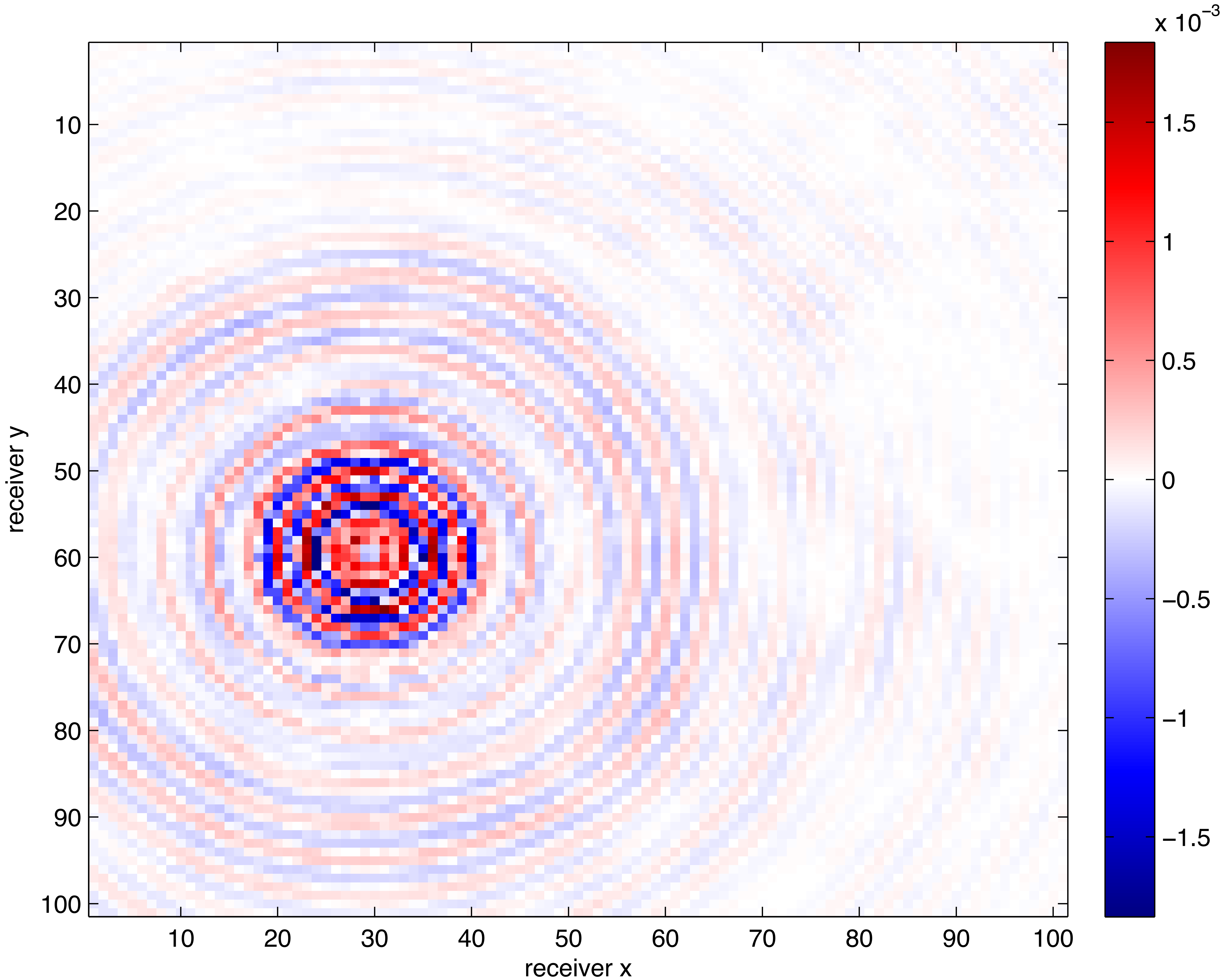


True data

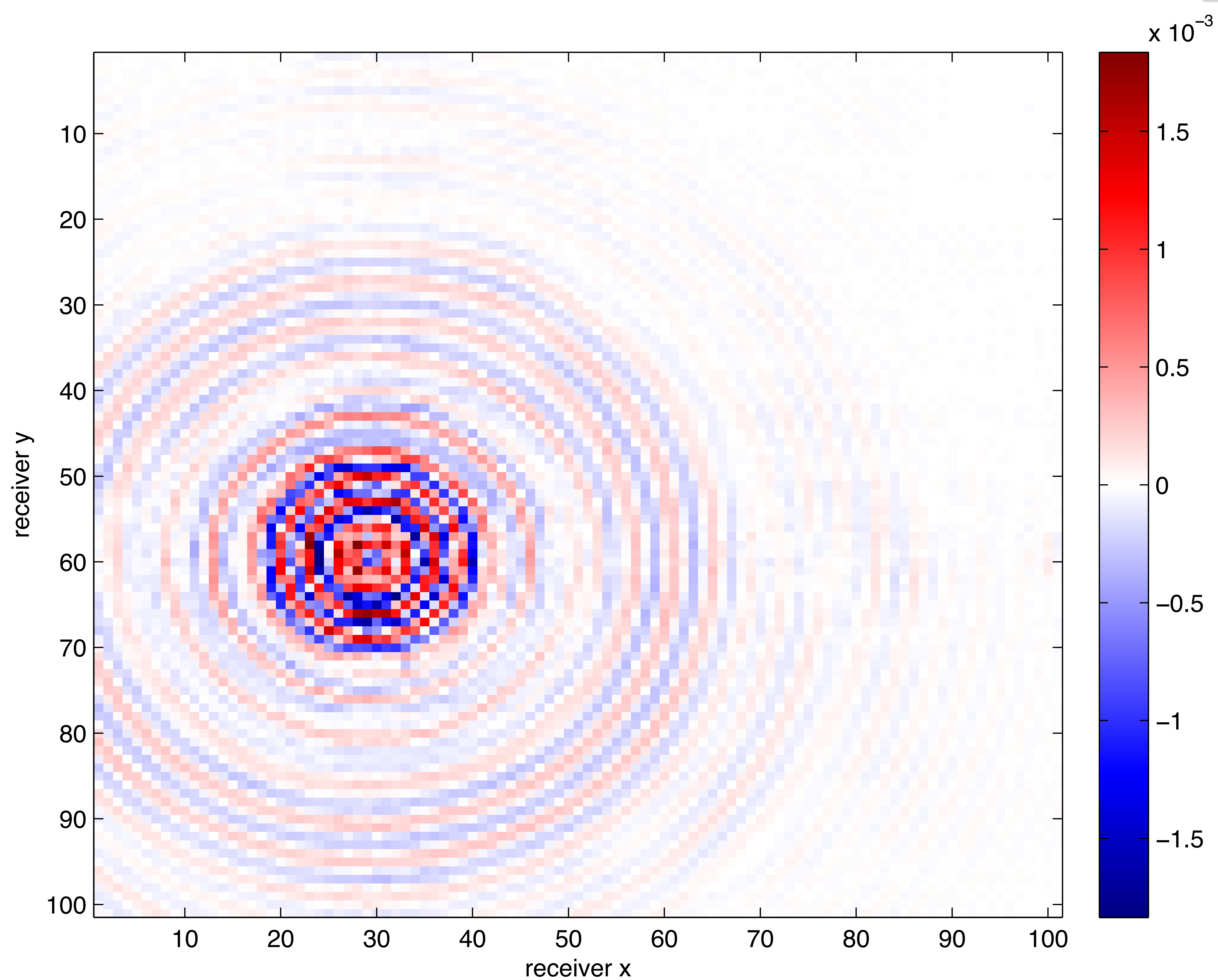


Subsampled data

Regularized recovery - 50% missing receivers

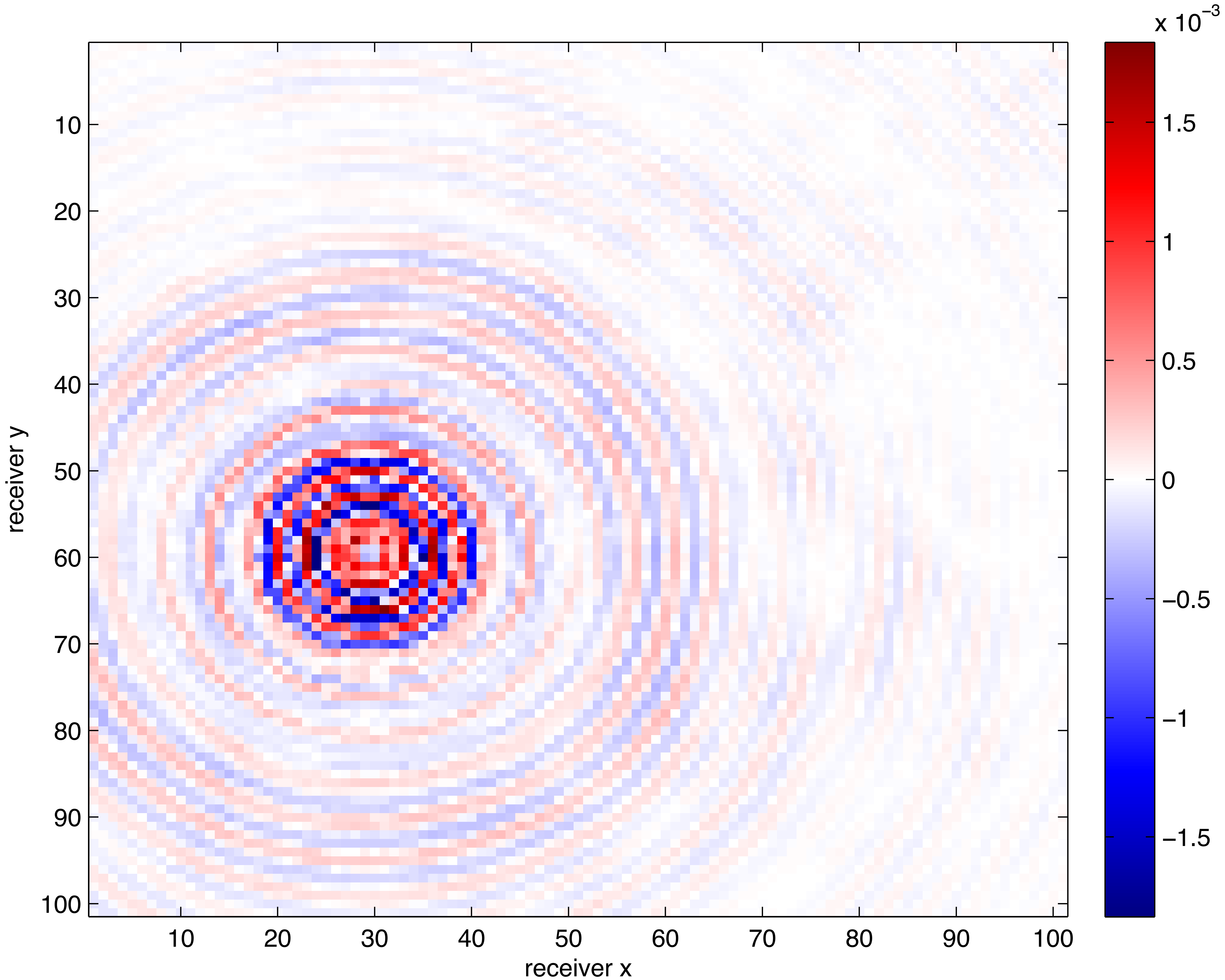


True data

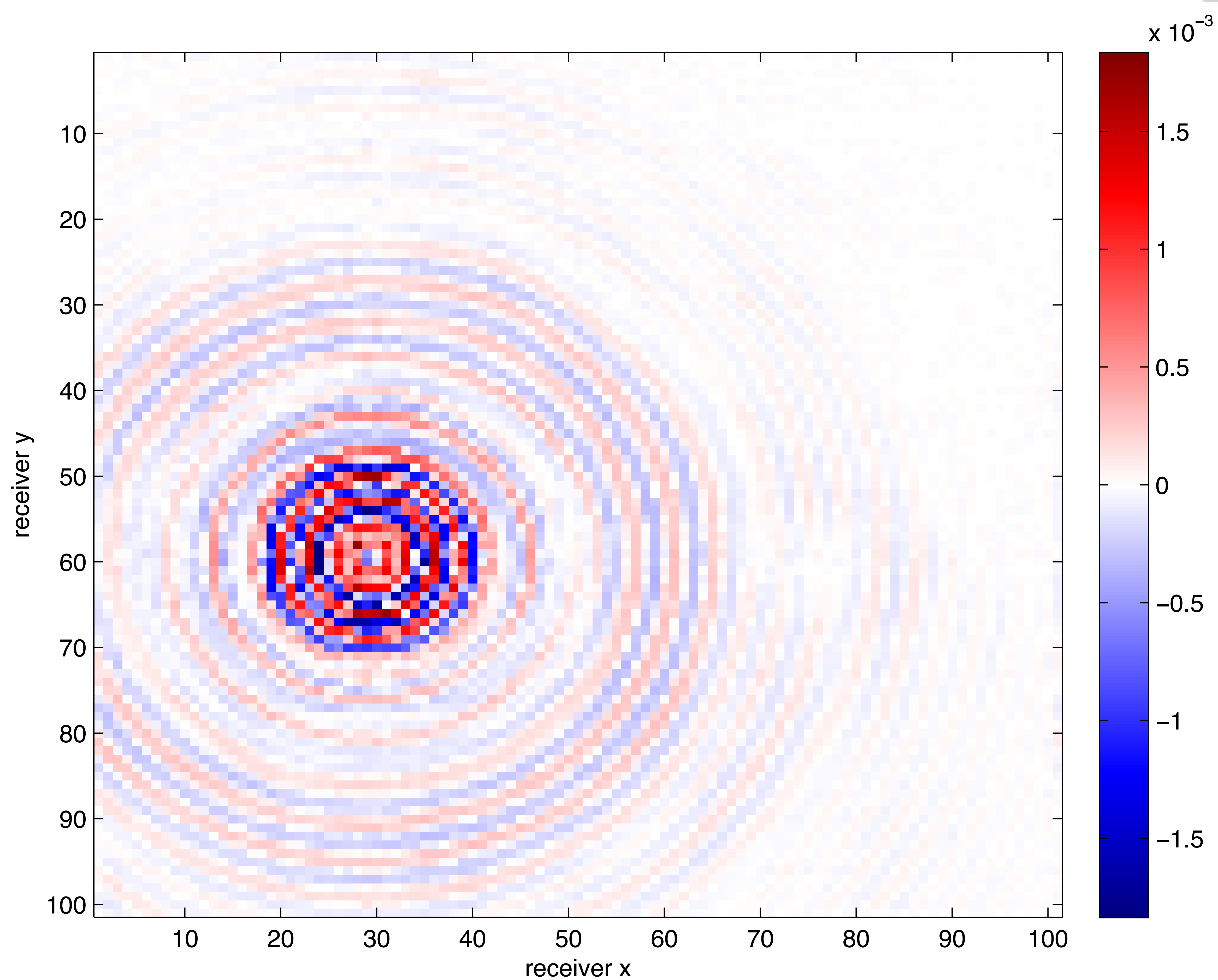


Without regularization - SNR 9.71 dB

Regularized recovery - 50% missing receivers

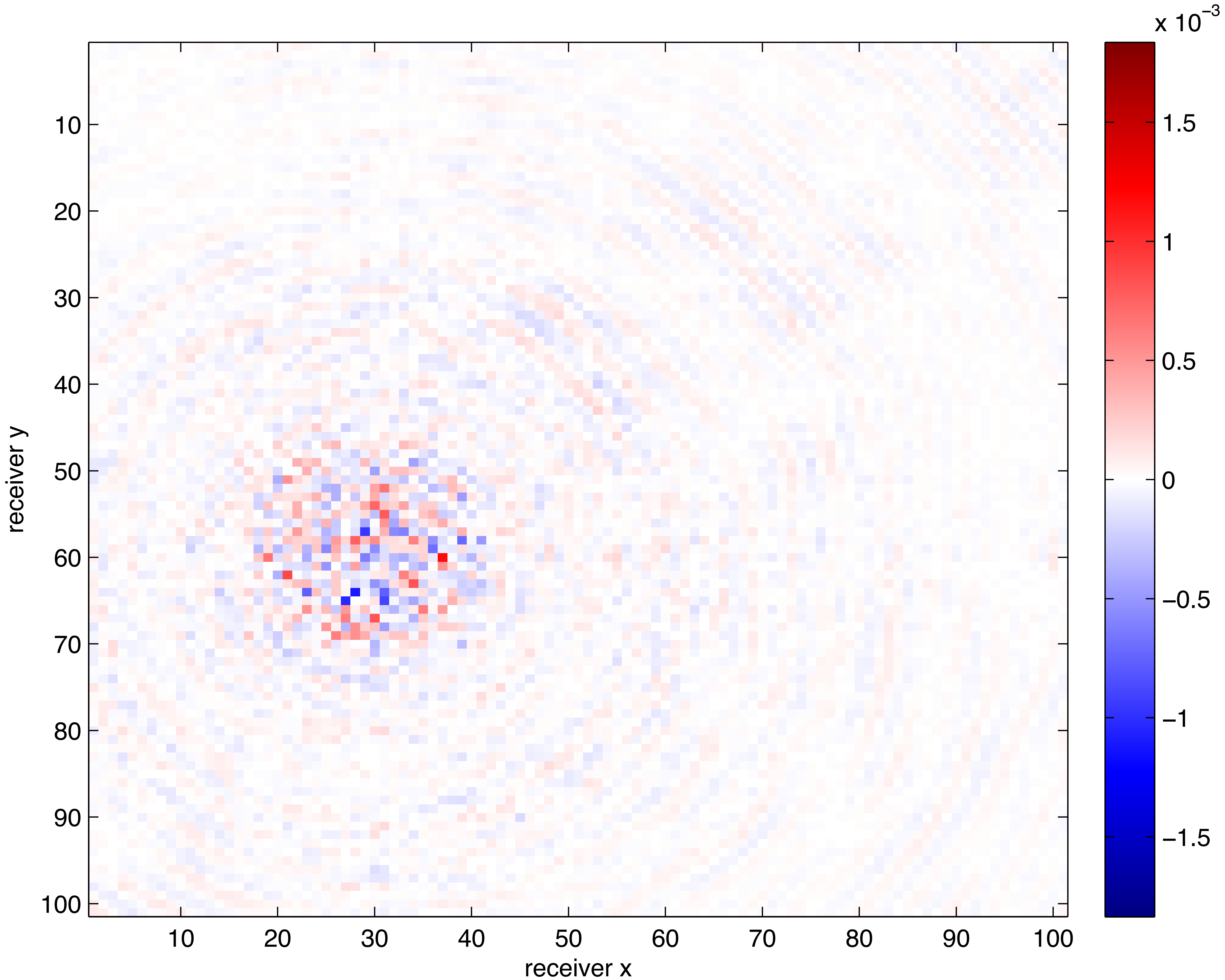


True data

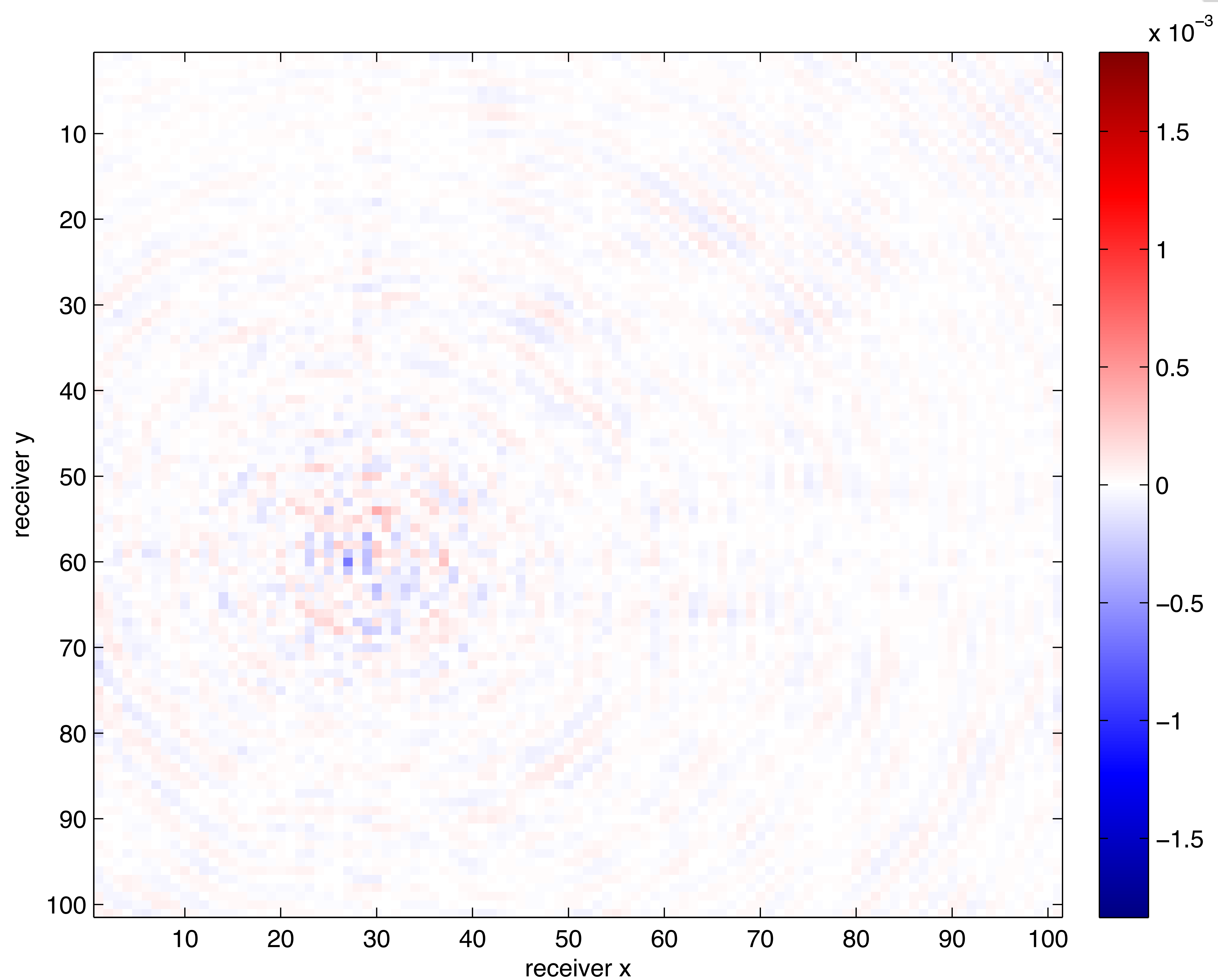


With regularization - SNR 15.3 dB

Regularized recovery - 50% missing receivers



Difference - no regularization



Difference - regularization

Summary

Irregular sampling destroys low-rank behaviour

- need an appropriate transform to operate in a low-rank domain

Regularization improves interpolation results

- can be easily incorporated in to optimization framework

Need for a fast interpolation transform

- more research needed

Summary

HTOpt

- previously released software for tensor interpolation

Software releases of SPGLR, updated HTOpt to come soon*

*graduate student clocks may not be synced to global clocks

Acknowledgements

Thank you for your attention

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, CGG, Chevron, ConocoPhillips, ION, Petrobras, PGS, Statoil, Total SA, Sub Salt Solutions, WesternGeco, and Woodside.