To redraw or *not* to redraw: *recent* insights in *randomized* dimensionality reduction for *inversion*

Felix J. Herrmann

SLIM

Seismic Laboratory for Imaging and Modeling
the University of British Columbia

Pass on the message: recent insights in *randomized* dimensionality reduction for *inversion*

Felix J. Herrmann

SLIM

Seismic Laboratory for Imaging and Modeling
the University of British Columbia

Goals

Move from '*correlation*' based *processing* to (robust) *inversion*

- multiple 'correlations' & 'convolutions'
- or applications of the Jacobian and it's adjoint

Use *randomized* dimensionality reduction to remove *prohibitive* computational *demands*.

Use *robust* statistics in the misfit functionals

to allow for *outliers*.

Special circumstances

We are typically 'data *rich*' rather than 'data *poor*' to the point

- that data is *overwhelming* our *systems*
- this is a becoming an *impediment* for wide-spread *adaption* of wave-equation based *inversion*

However, this '*data deluge*' also gives us an *unique* possibility to come up with *extremely* fast *algorithms* that

- work on *small* subsets solving "*denoising*" problems
- are based on *large-system approximations* (*statistical physics*)

SLIM

Dimensionality reduction

Seismic imaging & inversion:

- linear in the sources
- cost dominated by # PDE solves

Exploit this property by working on much smaller *randomized* subsets (mini batches) of *source* experiments

Control errors by

- *nonlinear* transform-domain *sparsity* promotion a la CS
- *averaging* by *growing* the *batch* size a la SAA

Disclaimer

Our problems are *large* for which it is *extremely* challenging to

- *verify* conditions that *guarantee* recovery
- *converge* to solutions in *reasonable* compute *time*

Our *claims* of *actually* solving *optimization* problems has to be taken with a *grain* of *salt*... But, *not* all is lost because there exists a whole *body* of *heuristics*. Today's talk aims to

- *connect* perspectives (e.g. CS vs Stochastic optimization)
- gain *understanding* why we may be 'lucky'

Wave-equation migration

Solution of a large *'overdetermined'* system

$$\min_{\mathbf{x}} \frac{1}{2K} \sum_{i=1}^{K} \|\mathbf{b}_i - \mathbf{A}_i \mathbf{x}\|_2^2,$$

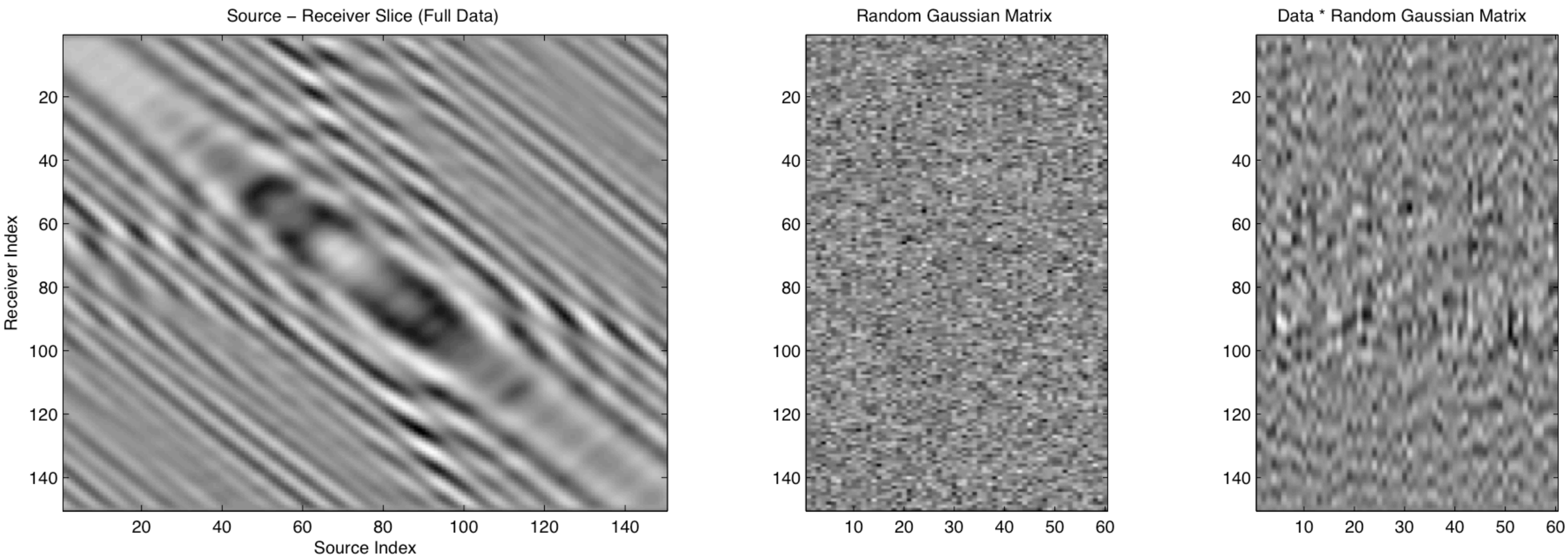
Iterations of the solver requires 4 PDE solves for each *source*

- use *linearity* of the *source* to turn *sequential* sources into *random* simultaneous / selected sources
- use *fewer* sources

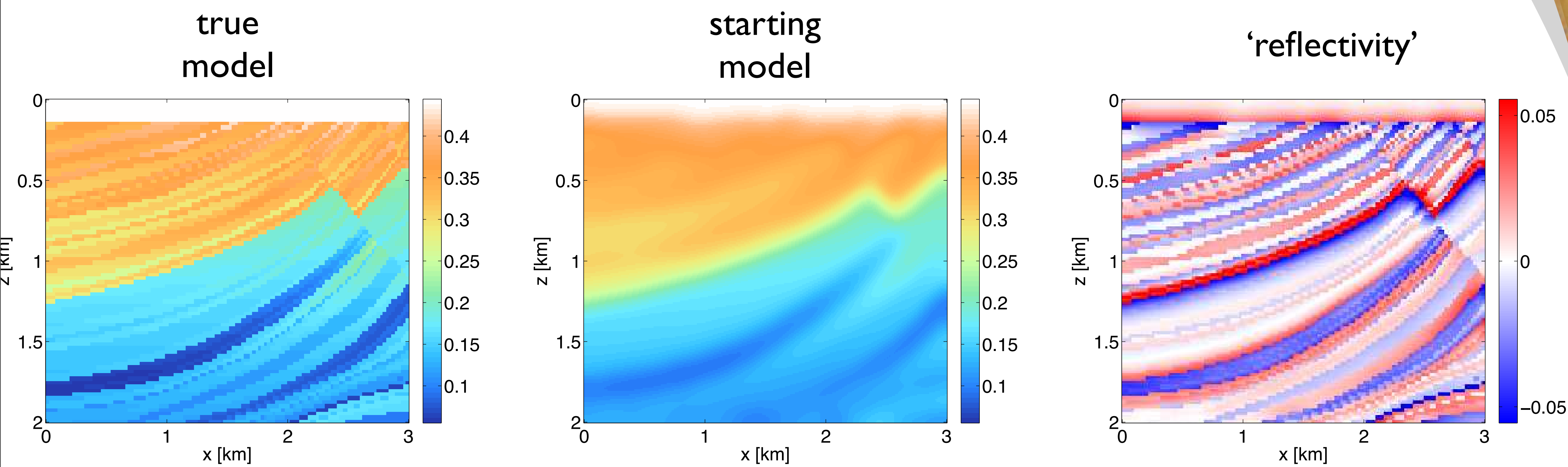
Study behavior as # of sources *increases*

Randomized source superposition

$$[\mathbf{b}_1, \cdots, \mathbf{b}_{n_s}]$$ $$\mathbf{W}$$ $$[\underline{\mathbf{b}}_1, \cdots, \underline{\mathbf{b}}_{n'_s}]$$



Stylized example



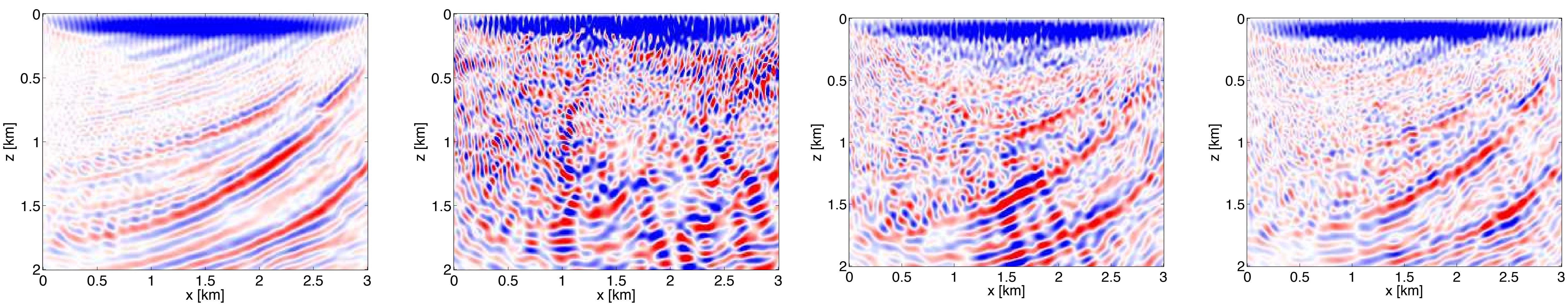
[Haber, Chung, and FJH, '10; van Leeuwen, Aravkin, FJH, '10]

SLIM

Migration

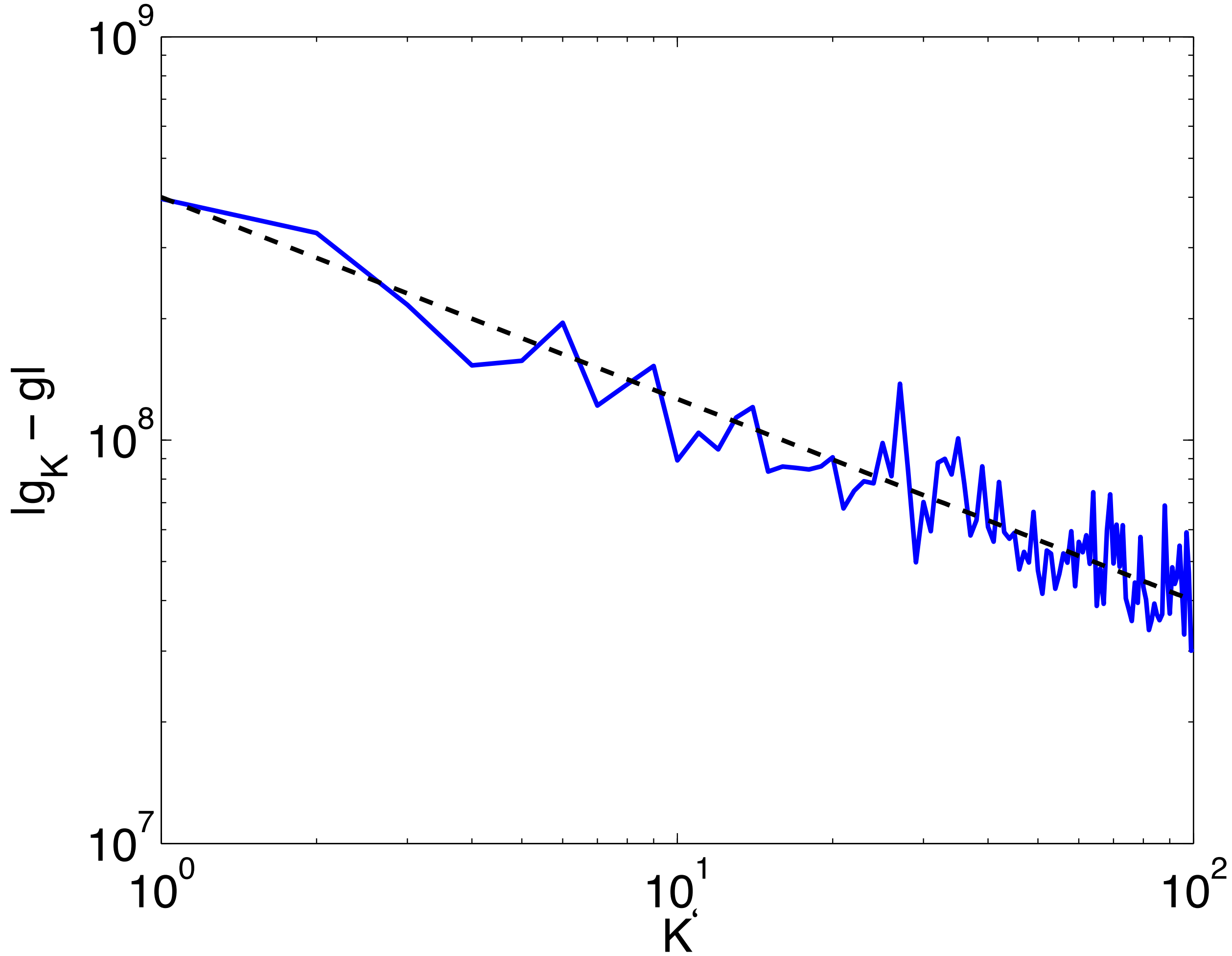
Search direction for *increasing* batch size K:

$$\mathbf{g}_{K'} \approx \frac{1}{K'} \sum_{j=1}^{K'} \mathbf{A}_j^* \underline{\mathbf{b}}_j \qquad K' = n'_f \times n'_s$$



full K'=1 K'=5 K'=10

Decay



error between full and sampled migration

[Haber & FJH, '10, van Leeuwen, '11, FJH et. all. '10-'11]

SLIM

Heuristics

Algorithm 1: Stochastic-average approximation with warm restarts

$\mathbf{x}_0 \longleftarrow \mathbf{0}; \mathbf{k} \longleftarrow \mathbf{0}$; `// initialize`

while $\|\mathbf{x}_0 - \widetilde{\mathbf{x}}\|_2 \geq \epsilon$ **do**

- $k \longleftarrow k + 1$; `// increase counter`
- $\widetilde{\mathbf{x}} \longleftarrow \mathbf{x}_0$; `// update warm start`
- $\mathbf{W} \longleftarrow \mathrm{Draw}(\mathbf{W})$; `// draw new subsampler`
- $\mathbf{x}_0 \longleftarrow \mathrm{Solve}(\mathbb{P}(\mathbf{W}); \widetilde{\mathbf{x}})$; `// solve the subproblem`

end

Subproblems

least-squares migration

$$\mathbb{P}_{\ell_2}(\mathbf{W}^k;\mathbf{x}_0):\quad \min_{\mathbf{x}} \frac{1}{2K'}\sum_{j=1}^{K'} \|\underline{\mathbf{b}}_j^k - \mathbf{A}_j^k\mathbf{x}\|_2^2$$

- solve with *limited* # of iterations of LSQR
- initialize solver with *warm* start
- solves *damped* least-squares problem

Subproblems

sparsity-promoting migration

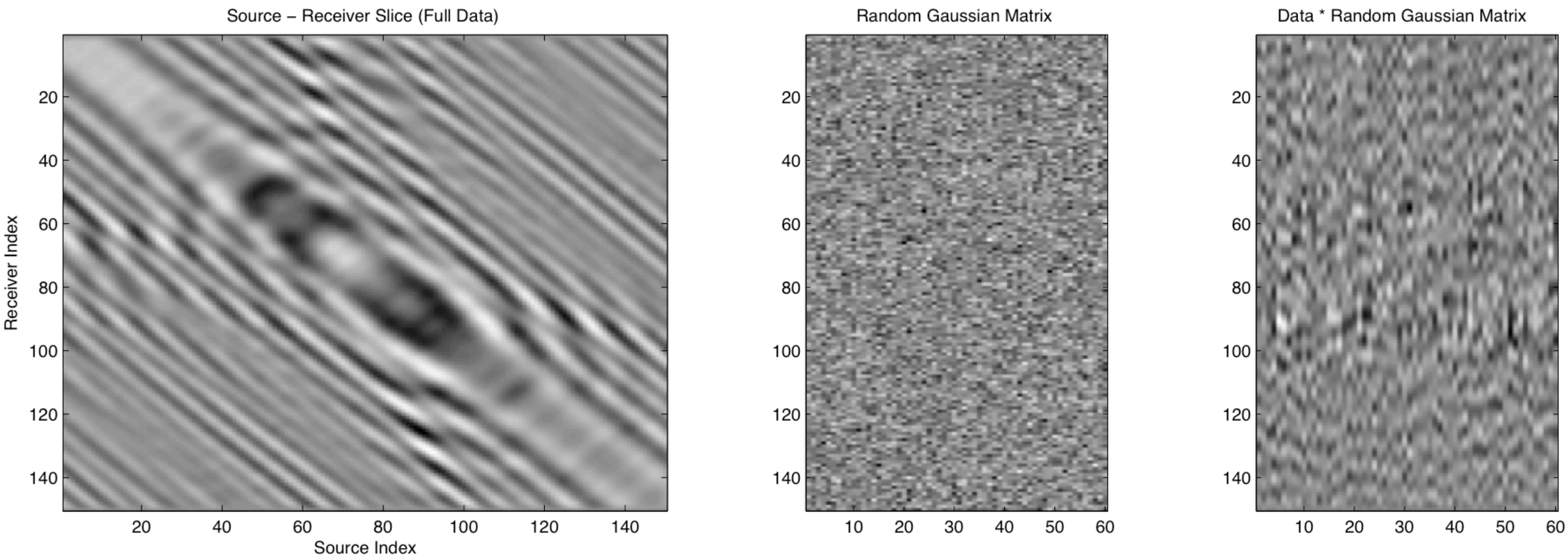
$$\mathbb{P}_{\ell_1}(\mathbf{W}^k;\mathbf{x}_0) \quad \min_{\mathbf{x}} \frac{1}{2K'}\sum_{j=1}^{K'} \|\underline{\mathbf{b}}_j^k - \mathbf{A}_j^k\mathbf{x}\|_2 \quad \text{subject to} \quad \|\mathbf{x}\|_{\ell_1} \leq \tau^k$$

- solve LASSO problem for a given *sparsity* level using the *spectral-gradient* method ($\mathrm{SPG}\ell_1$)
- initialize solver with *warm* start
- solves *sparsity-promoting* subproblem

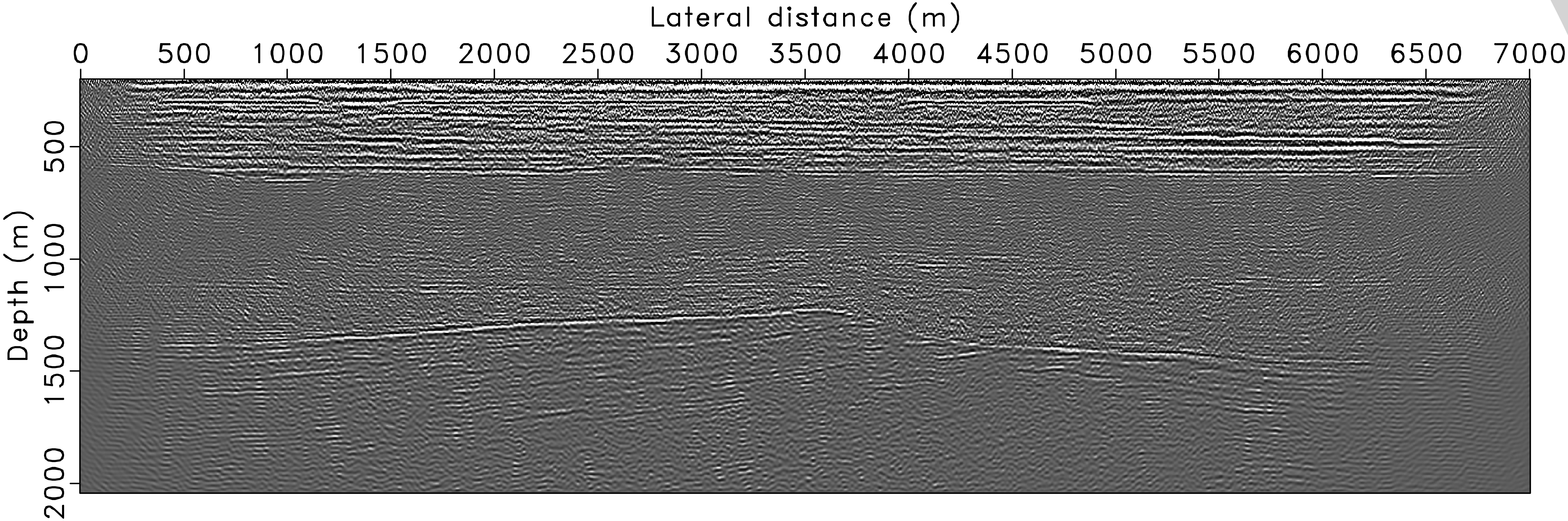
[van den Berg & Friedlander, '08]

Randomized source superposition

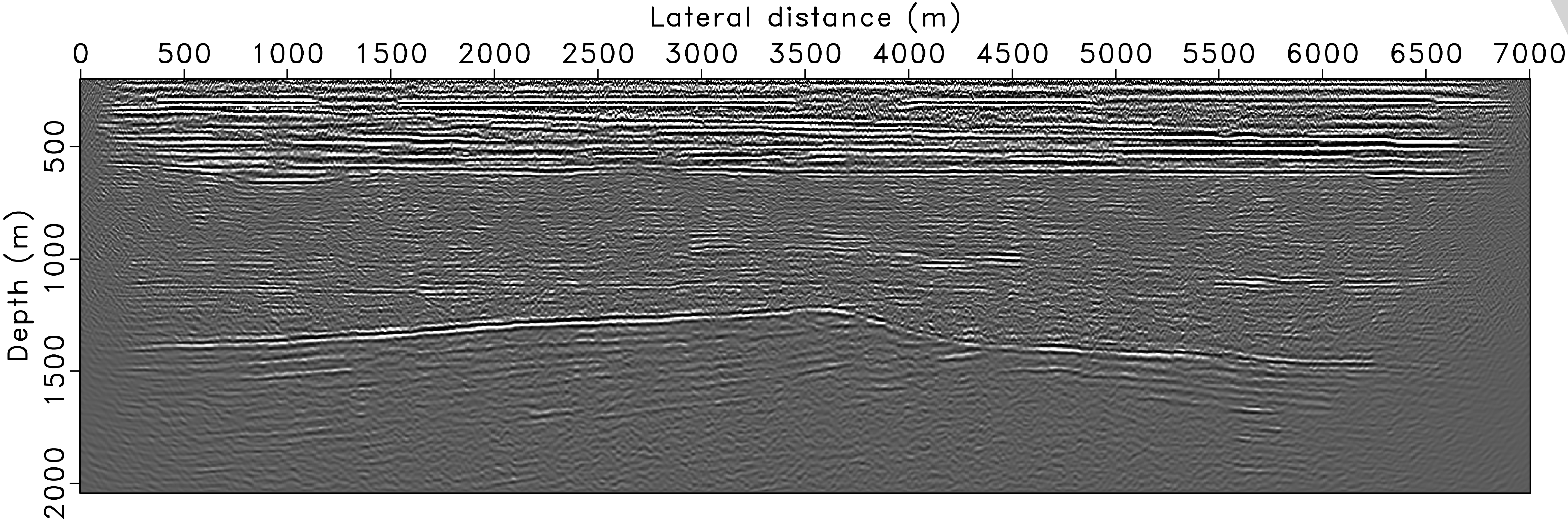
$$[\mathbf{b}_1, \cdots, \mathbf{b}_{n_s}] \qquad \mathbf{W} \qquad [\underline{\mathbf{b}}_1, \cdots, \underline{\mathbf{b}}_{n'_s}]$$



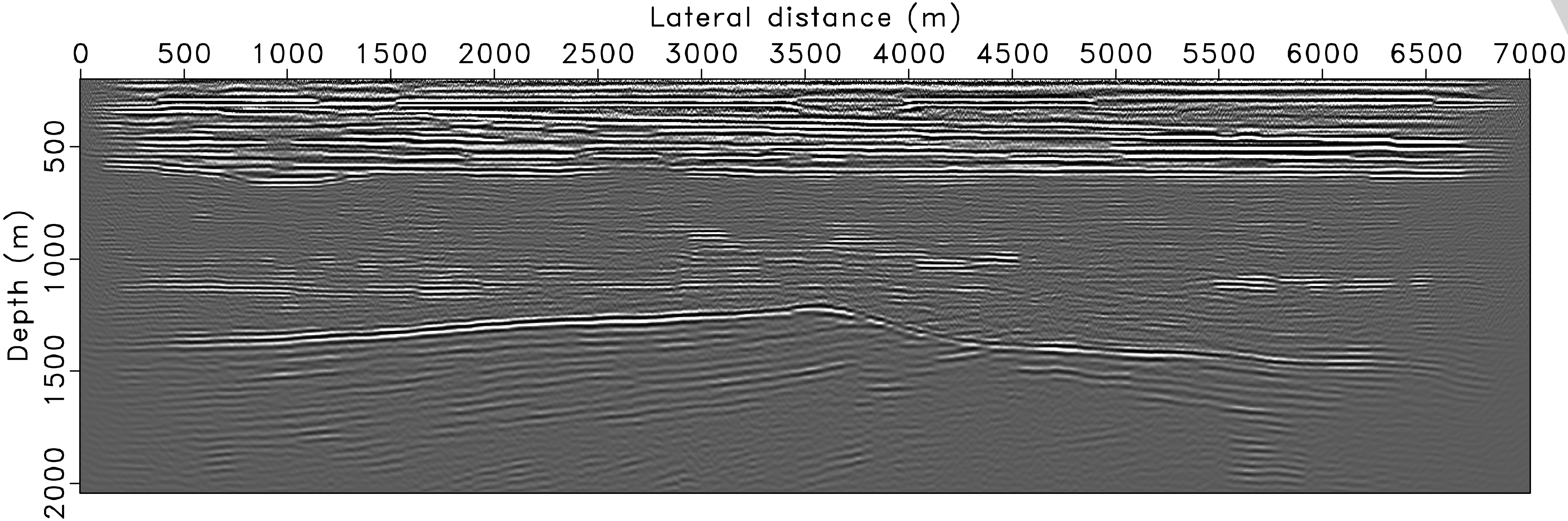
Least-squares migration



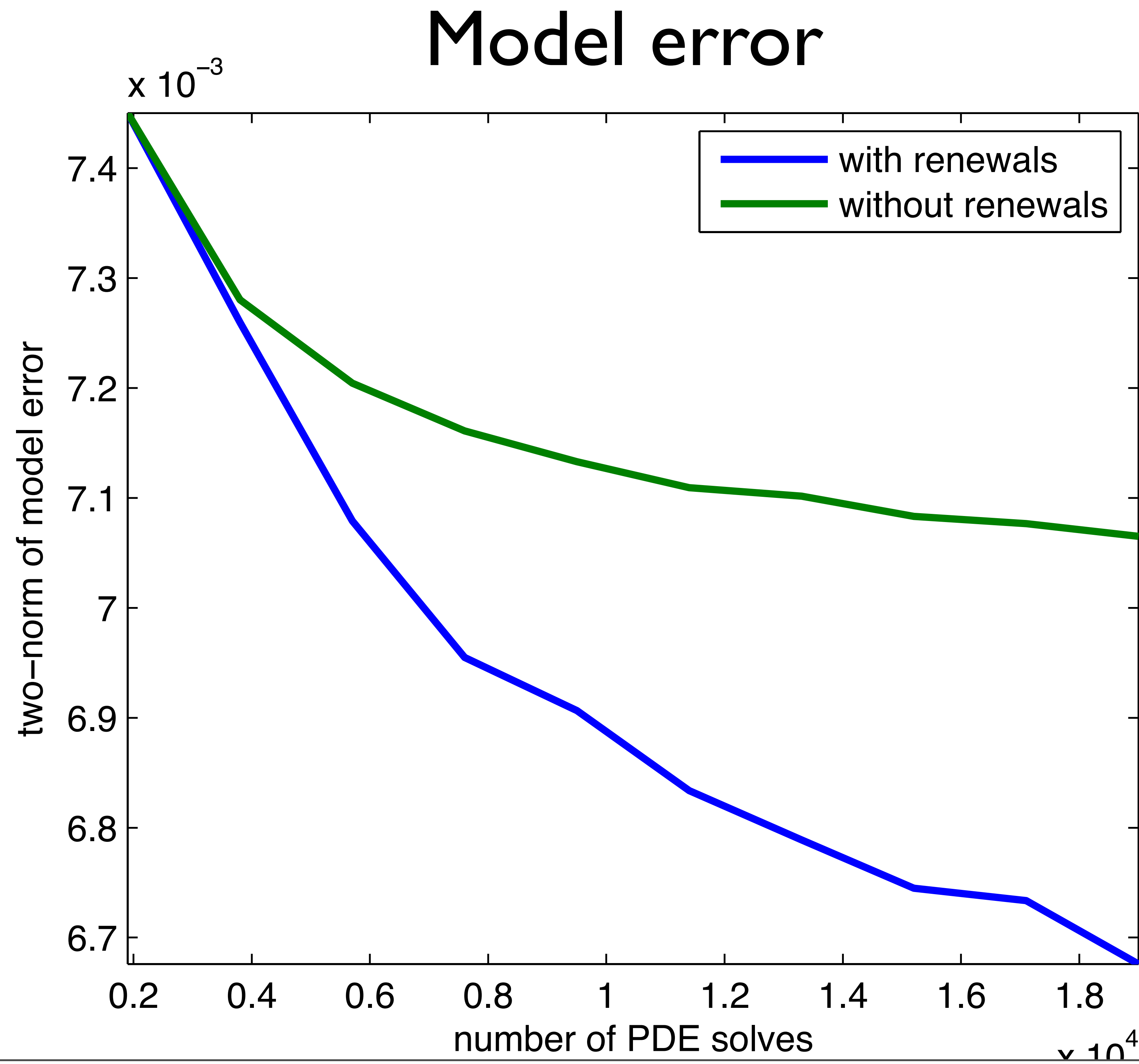
Sparsifying migration
without renewals



Sparsifying migration
with renewals



Sparsifying migration



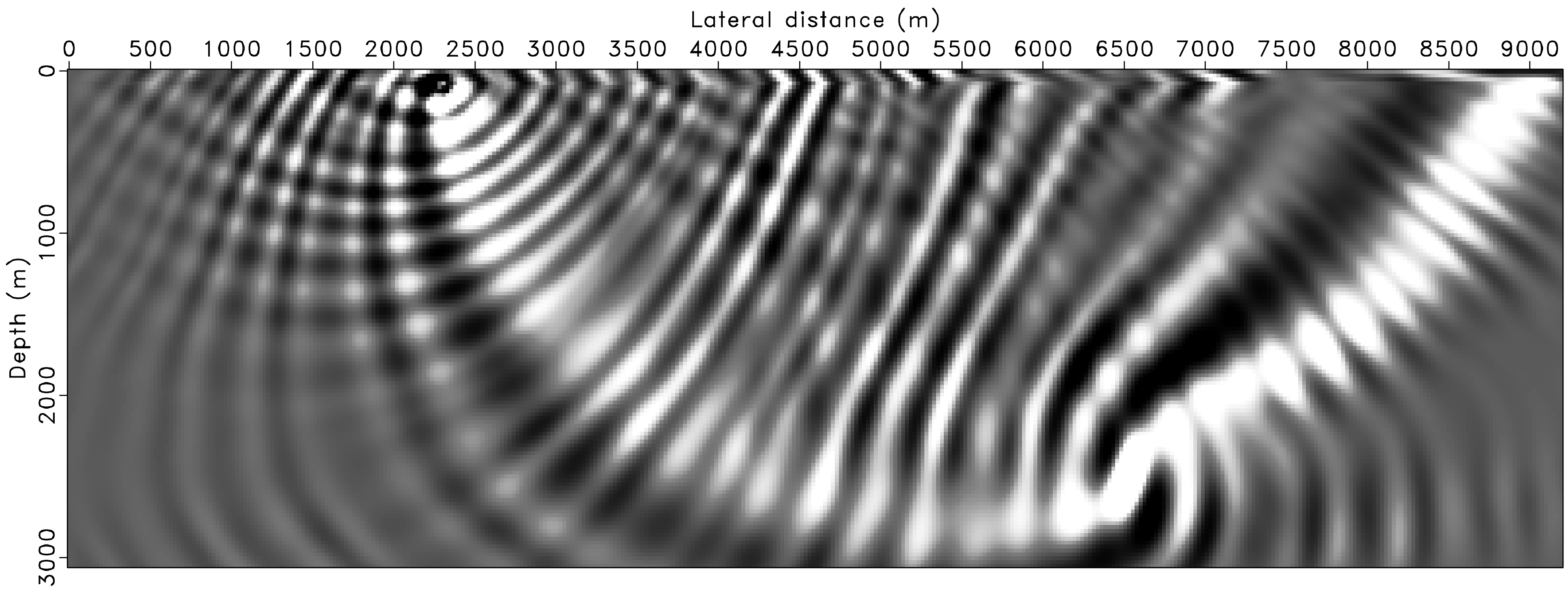
Why does this work?

Geophysics perspective:

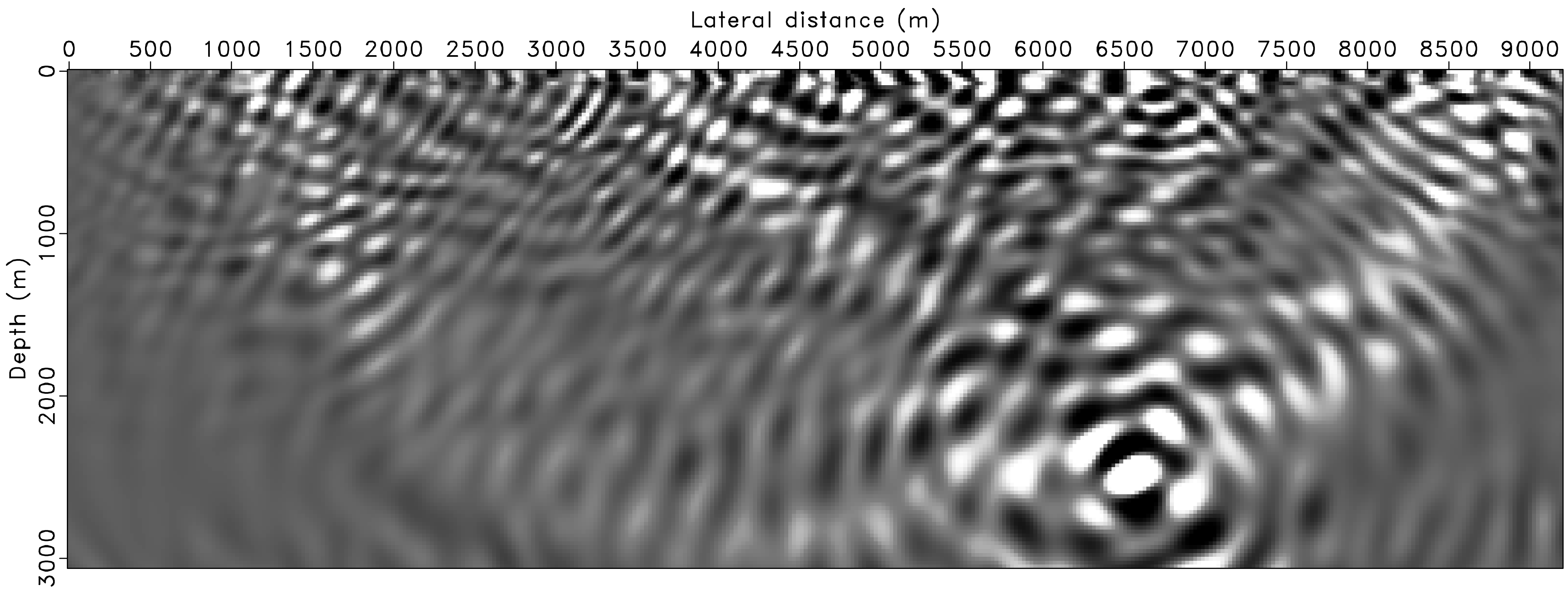
- *richer* wavenumber *content* of the *randomized* simultaneous sources

This is the premise of '*phase encoding*'.

Image
from one shot

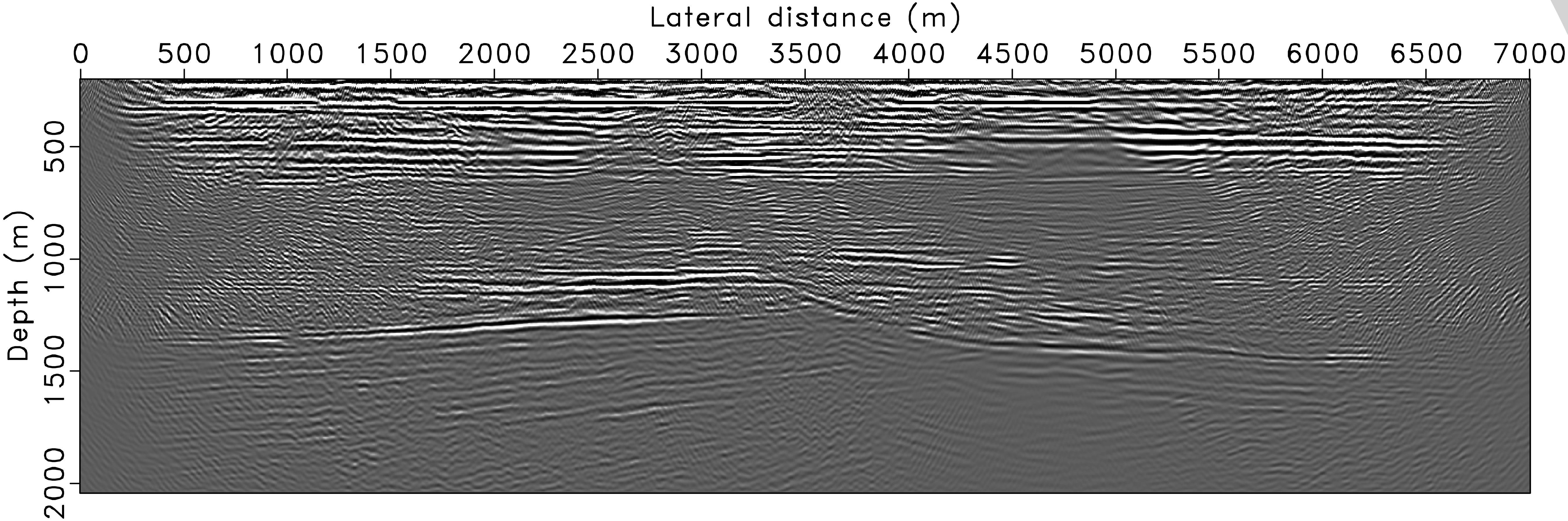


Sequential shot image



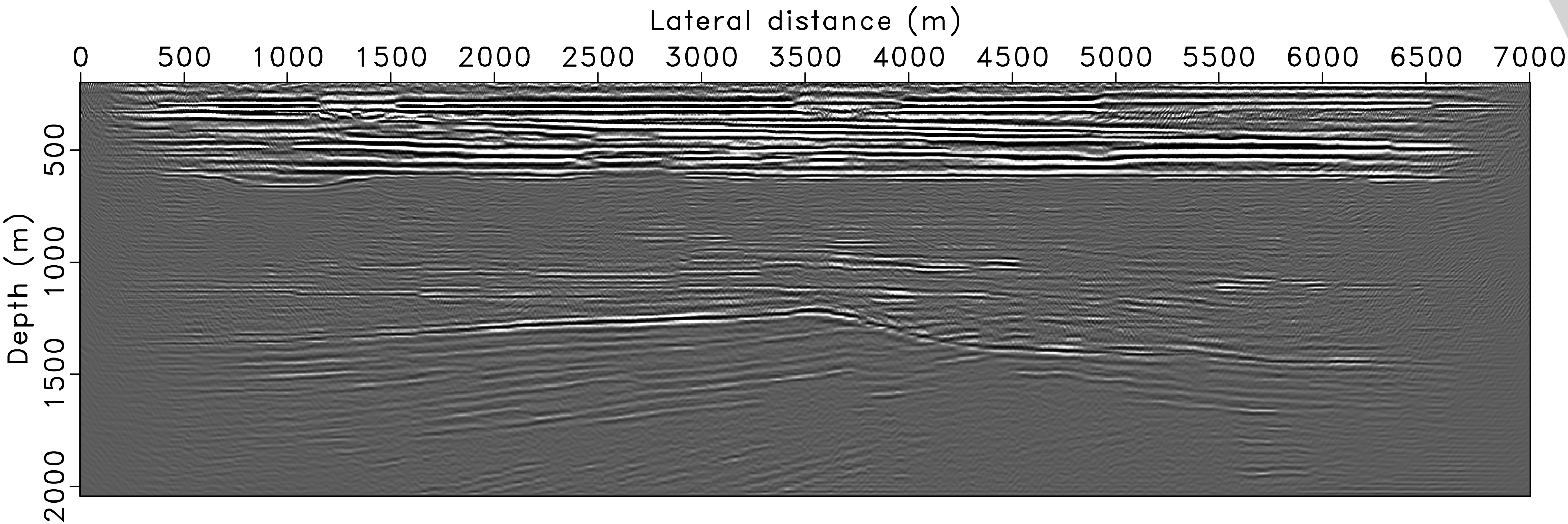
Simultaneous shot image

Sparsifying migration
without renewals



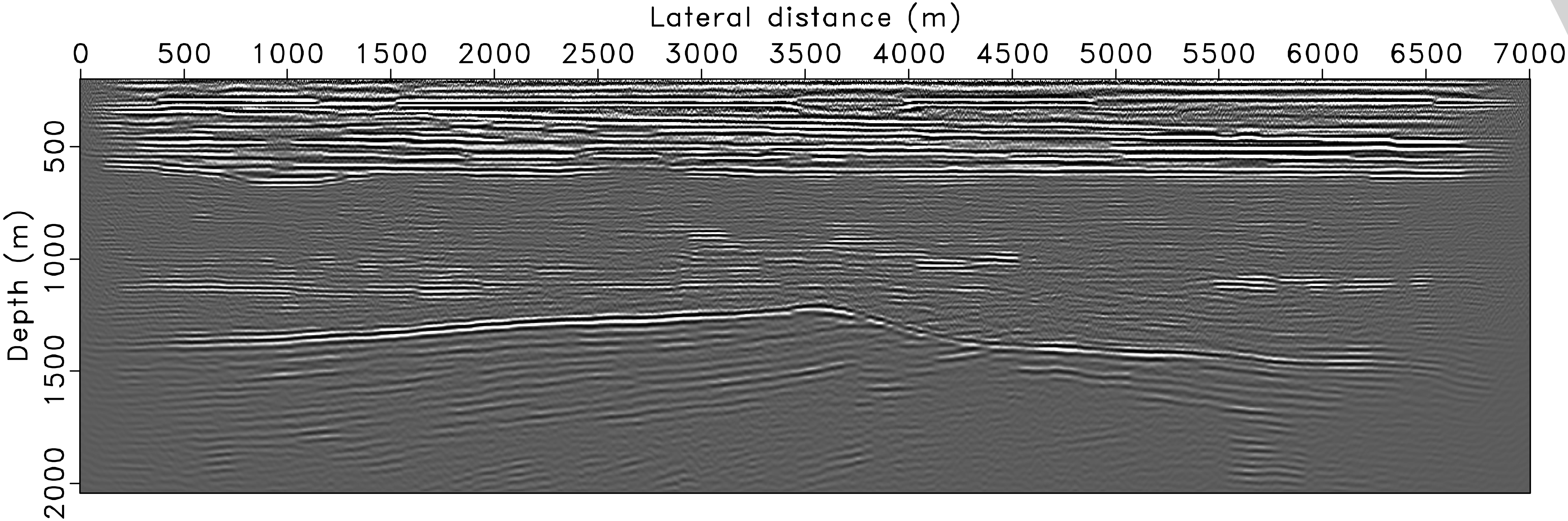
randomly selected *sequential* shots

Sparsifying migration
with renewals



randomly selected *sequential* shots

Sparsifying migration
with renewals



randomized *simultaneous* shots

Why does this work?

Geophysics perspective:

- *richer* wavenumber *content* of the *randomized* simultaneous sources

This is the premise of *'phase encoding*.

But this does *not* really explain why this also works for *randomly* selected impulsive shots...

Why does this work?

Inversion perspective:

- *sparsity* promotion acts as a *regularization*

This is the premise of Tikhonov regularization

Explains why inversion quality is *improved* but does *not* explain the *increased* decay of the model *error*...

Why does this work?

From the optimizer's perspective:

- aside from ideas from *stochastic* optimization *cooling* method are known to lead to *fast* algorithms

Combination of these *two* ideas may to be the way to go...

SLIM

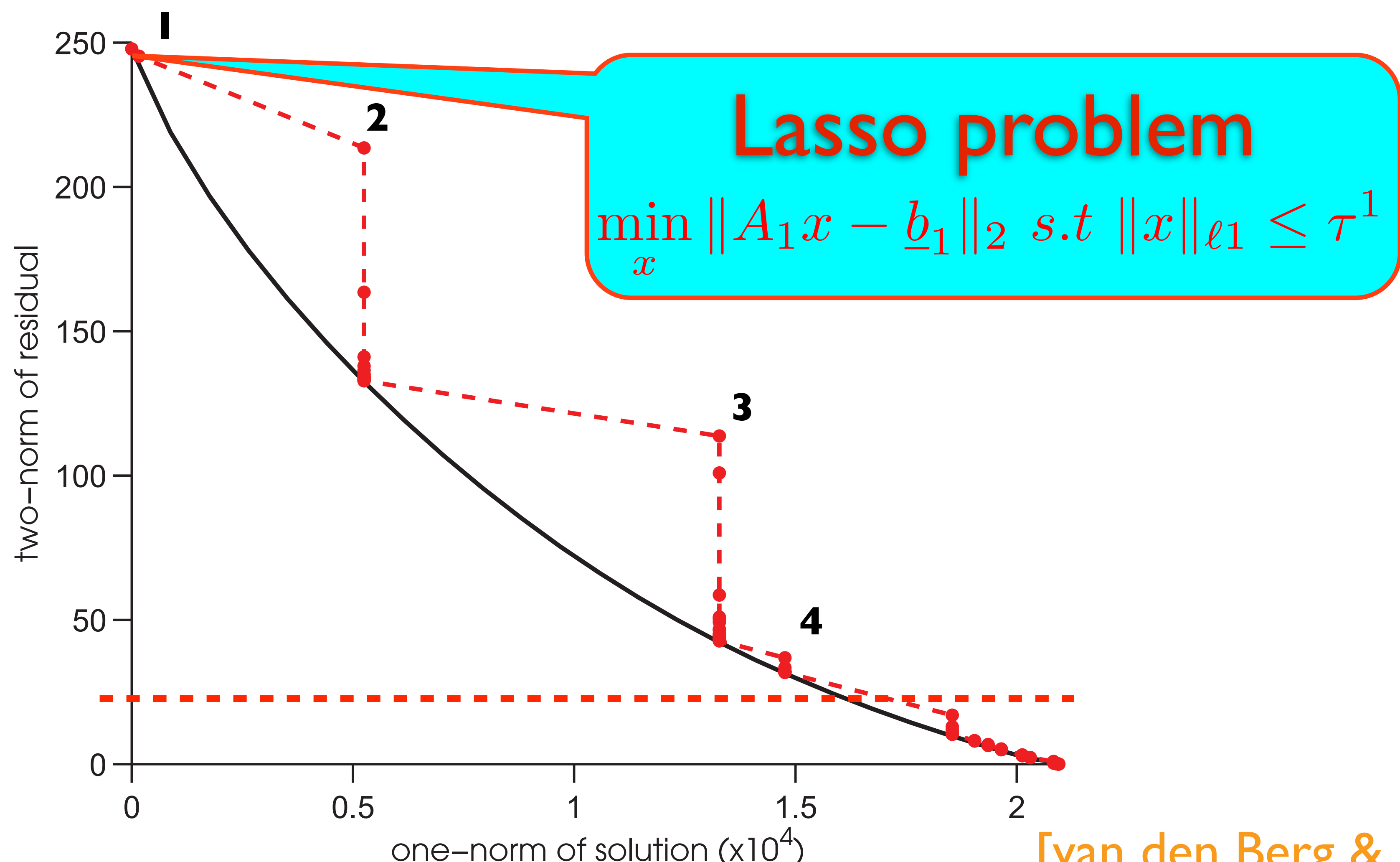
Continuation methods

Large-scale *sparsity*-promoting solvers *limit* the number of *matrix-vector* multiplies by

- slowly allowing *components* to *enter* into the *solution*
- solving an *intelligent* series of LASSO *subproblems* for *decreasing* sparsity levels
- exploring properties of the Pareto trade-off curve

Pareto curve

subproblems



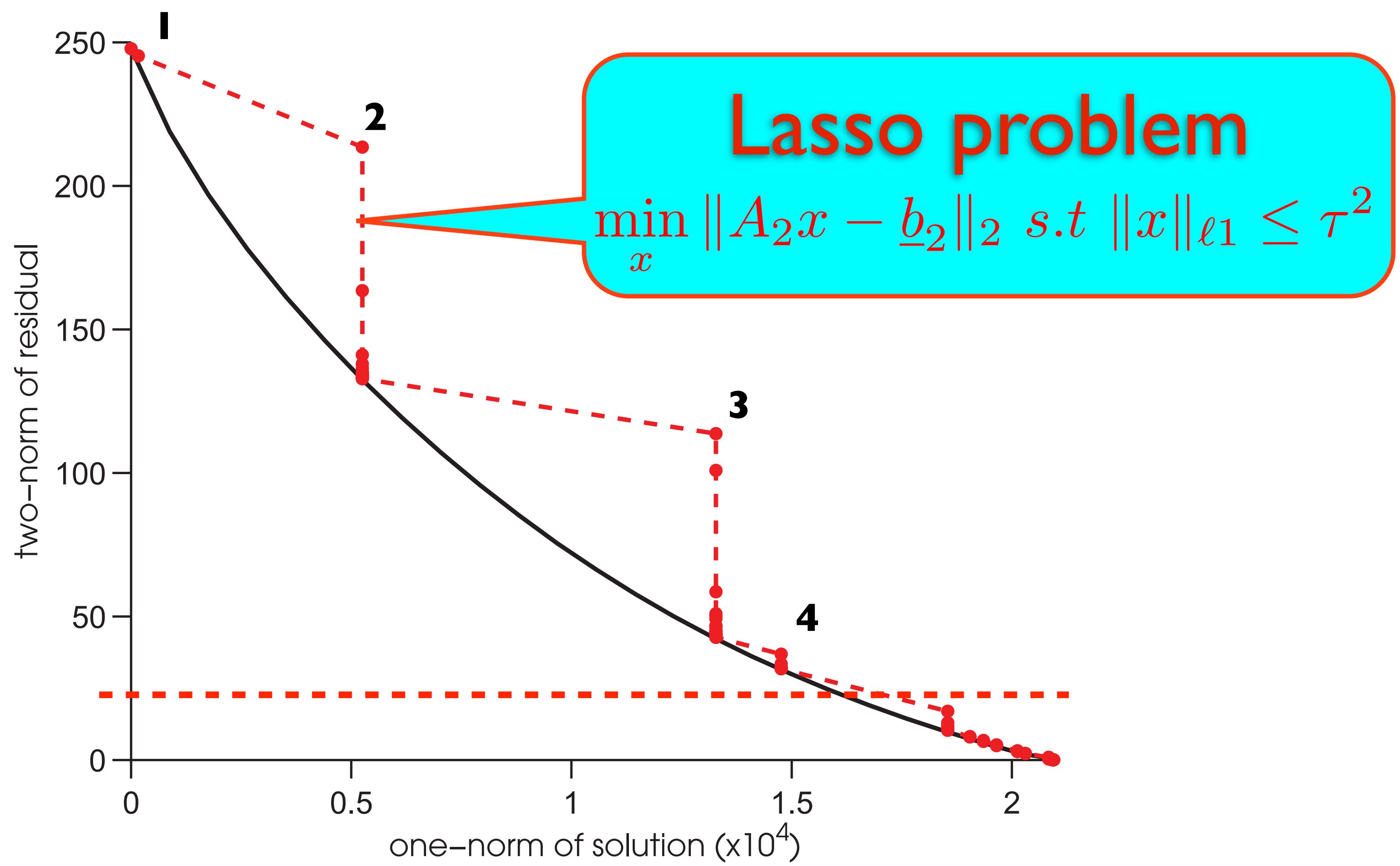
[van den Berg & Friedlander, '08]

[Hennefent et. al., '08]

[Lin & FJH, '09-]

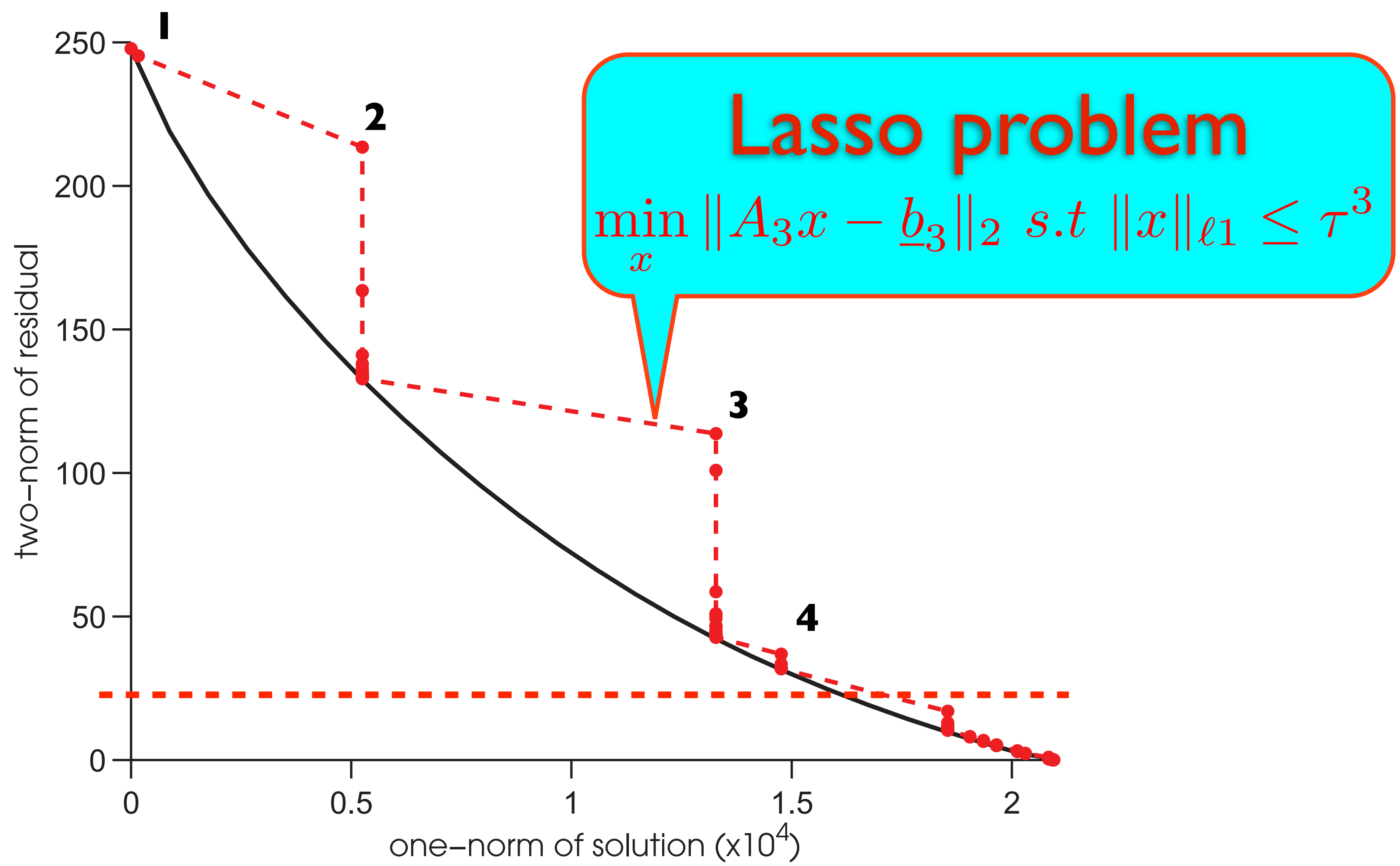
Pareto curve

subproblems



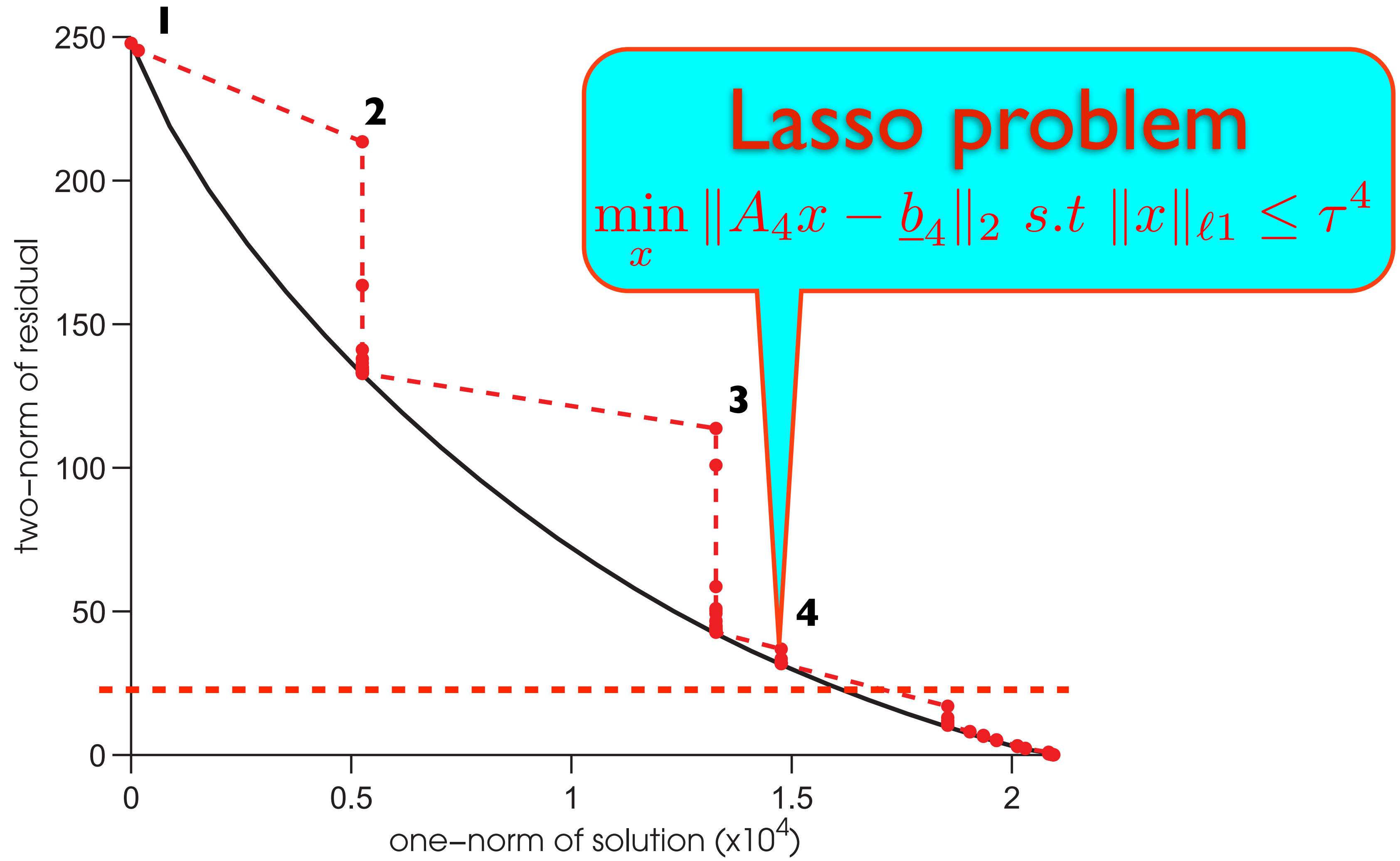
Pareto curve

subproblems



Pareto curve

subproblems



Why does this work?

Mathematics perspective:

- *randomization* makes sparsity-promoting program computationally *tractable*

This is the premise of *randomized* dimensionality reduction.

But again ideas from CS alone do *not* really explain the *improved* image *quality* with *renewals*.

So what's going on?

Why does this work?

Physicist's perspective:

We are dealing with *extremely* large *systems* that *mix* for

- *large* enough system *sizes* and long enough *times*
- *large* enough *complexity* in the *velocity* model

Linear systems start to behave like 'Gaussian' matrices

- show 'phase-transitions' for *simple* recovery *algorithms*
- *approximations* become *better* when systems get *larger*

[Daubechies et. al, ’04; Hennenfent et. al.,’08, Mallat, ’09, Donoho et. al, ’09]

Back to the oldies

Compressive sensing was all about designing *sampling* matrices that *create* white *Gaussian interferences*.

First iteration of *iterative* soft thresholding corresponds to vanilla *denoising*.

But does the *same* hold for later (t>1) *iterations* of

$$\mathbf{x}^{t+1} = \eta_t \left(\mathbf{A}^* \mathbf{z}^t + \mathbf{x}^t \right)$$
$$\mathbf{z}^t = \mathbf{b} - \mathbf{A}\mathbf{x}^t$$

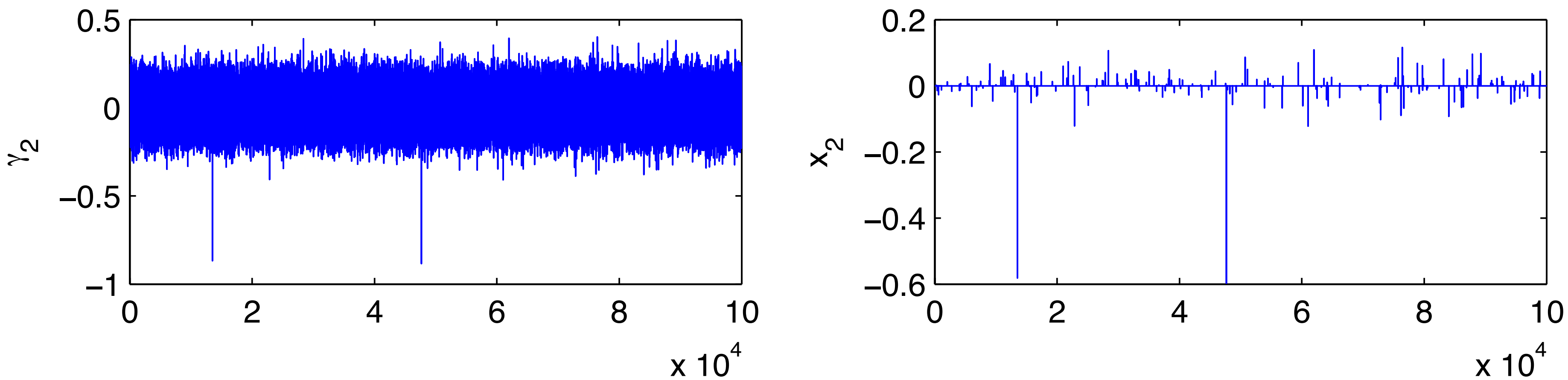
with threshold given by n^{th} largest coefficient of $\mathbf{A}^* \mathbf{z}^t + \mathbf{x}^t$

[http://sourceforge.net/projects/gampmatlab/files/gampmatlab20111128.zip]

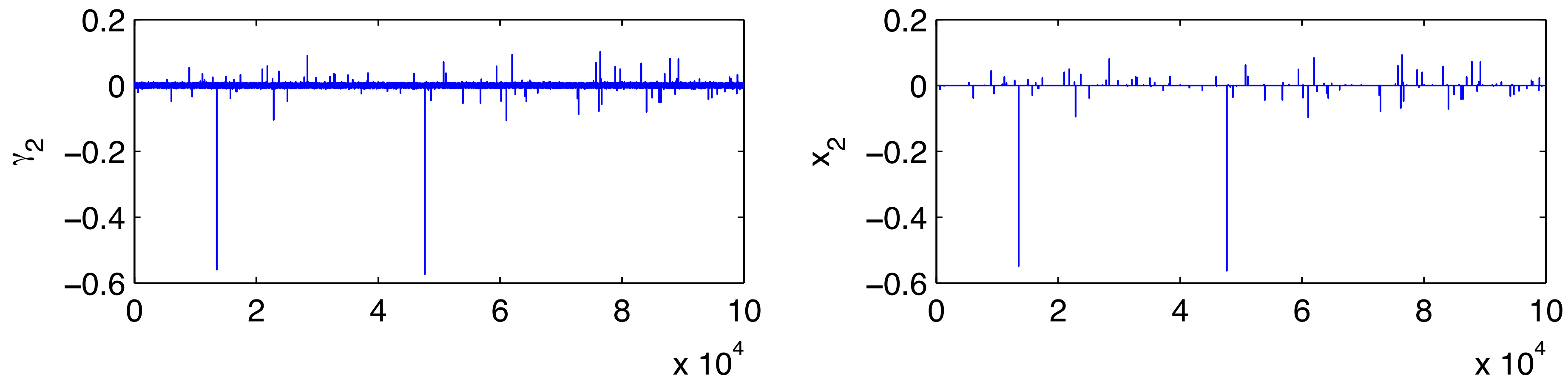
Setup

```
% Number of iterations of the algorithm
T = 10;

% Stopping criterion (tolerance for successfull decoding)
tol = 1e-4;
n = 200;
k = 2;
N = 100000;

A = (1/sqrt(n)) .* randn(n, N);

% Sparse signal (with uniform distribution of non-zeros)
x = [sign(rand(k,1) - 0.5); zeros(N-k,1)];
x = x(randperm(N));

% Generate Measurements
b = A*x;
xhat = reconstructAmp(A, b, T, tol,x,1);
```

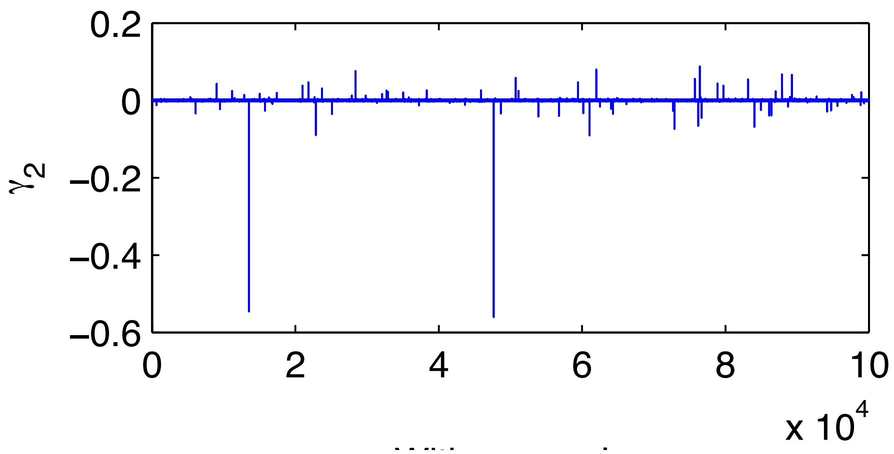
Iteration 1

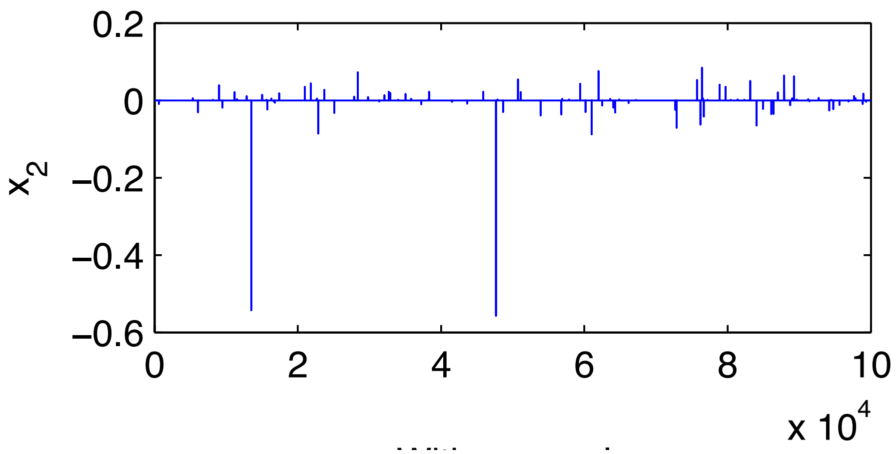


Iteration 2

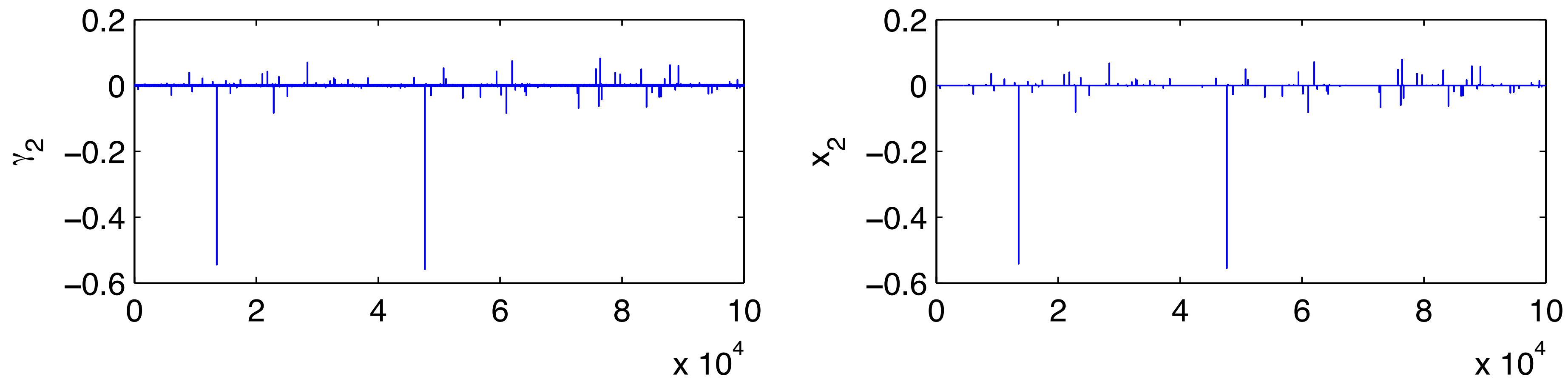


Iteration 3





Iteration 4



Problem

After *first* iteration the *inteferences* become 'spiky'

- *assumption* spiky vs white Gaussian *no* longer *holds*
- renders soft *thresholding* less *effective*

Leads to *slow* convergence of the *algorithm*.

Is there a way out?

Approximate message passing

Add a *term* to *iterative* soft *thresholding*, i.e.,

$$\mathbf{x}^{t+1} = \eta_t \left(\mathbf{A}^* \mathbf{z}^t + \mathbf{x}^t \right)$$

$$\mathbf{z}^t = \mathbf{b} - \mathbf{A}\mathbf{x}^t - \frac{1}{n}\mathbf{z}^{t-1} \sum \left(\eta' (\mathbf{A}^* \mathbf{z}^t + \mathbf{x}^t) \right)$$

with

$$\eta'(x) = \begin{cases} 1 & |x| > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Approximate message passing

According to Montanari the AMP algorithm corresponds to

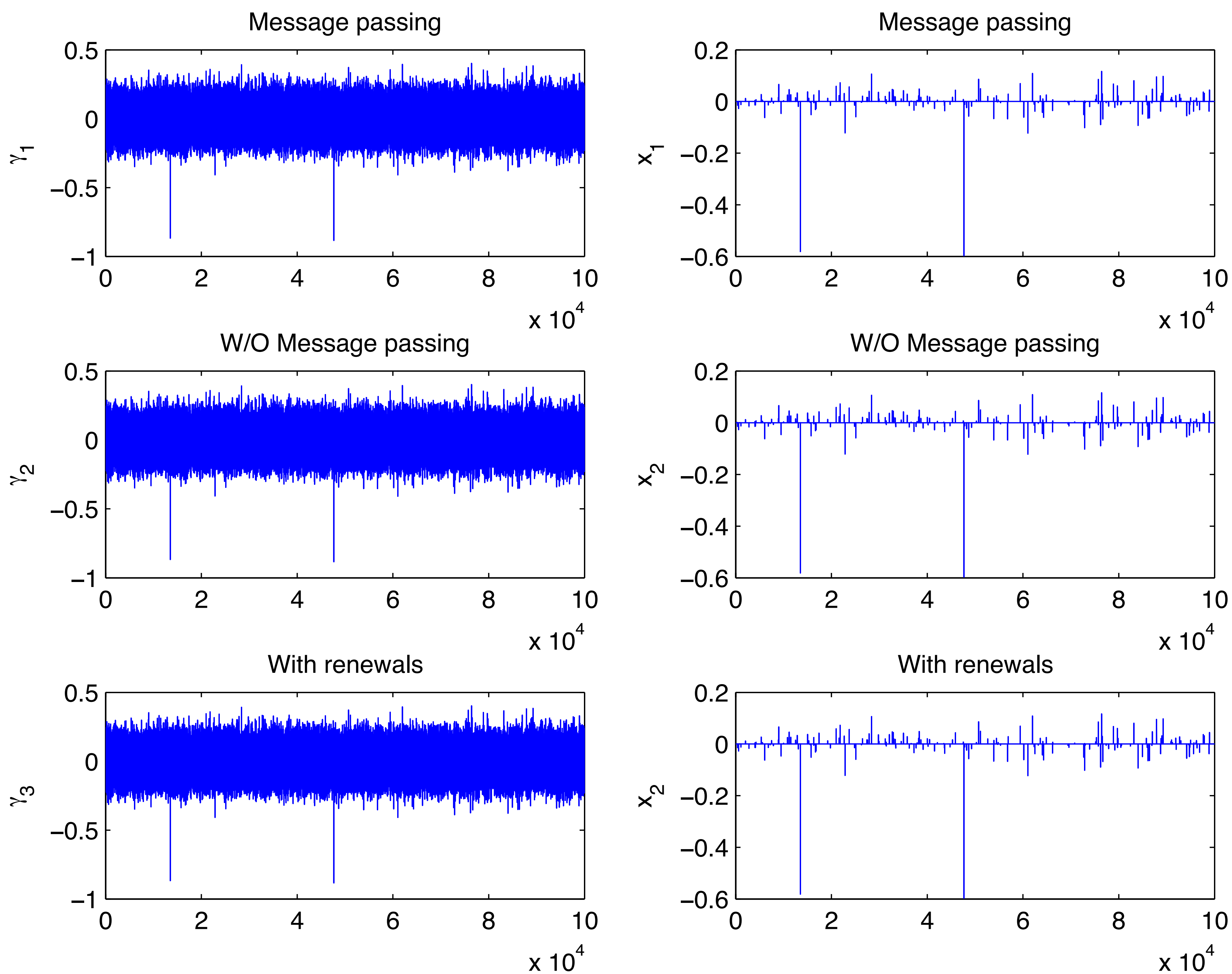
$$\mathbf{x}^{t+1} = \eta_t\left(\mathbf{A}_t^{*}\mathbf{z}^t + \mathbf{x}^t\right)$$
$$\mathbf{z}^t = \mathbf{b}_t - \mathbf{A}_t\mathbf{x}^t$$

where for *each* iteration a *new* CS *matrix* and *data* are *drawn.*

Changes the story *completely*

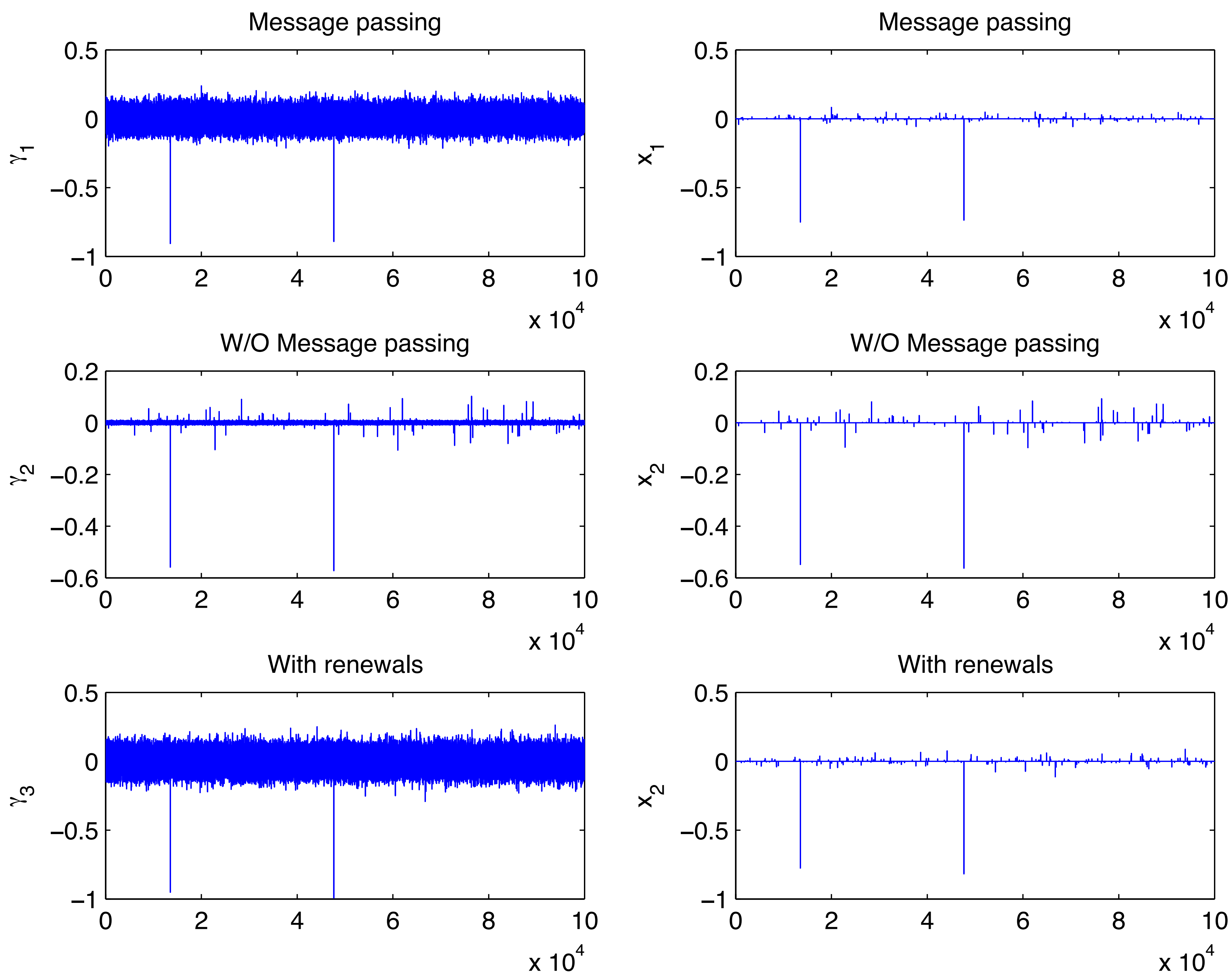
- draw *new* random subsets (e.g. shots) for *each* iteration
- *nonlinearity* improves the *performance* compared to SA

Iteration 1



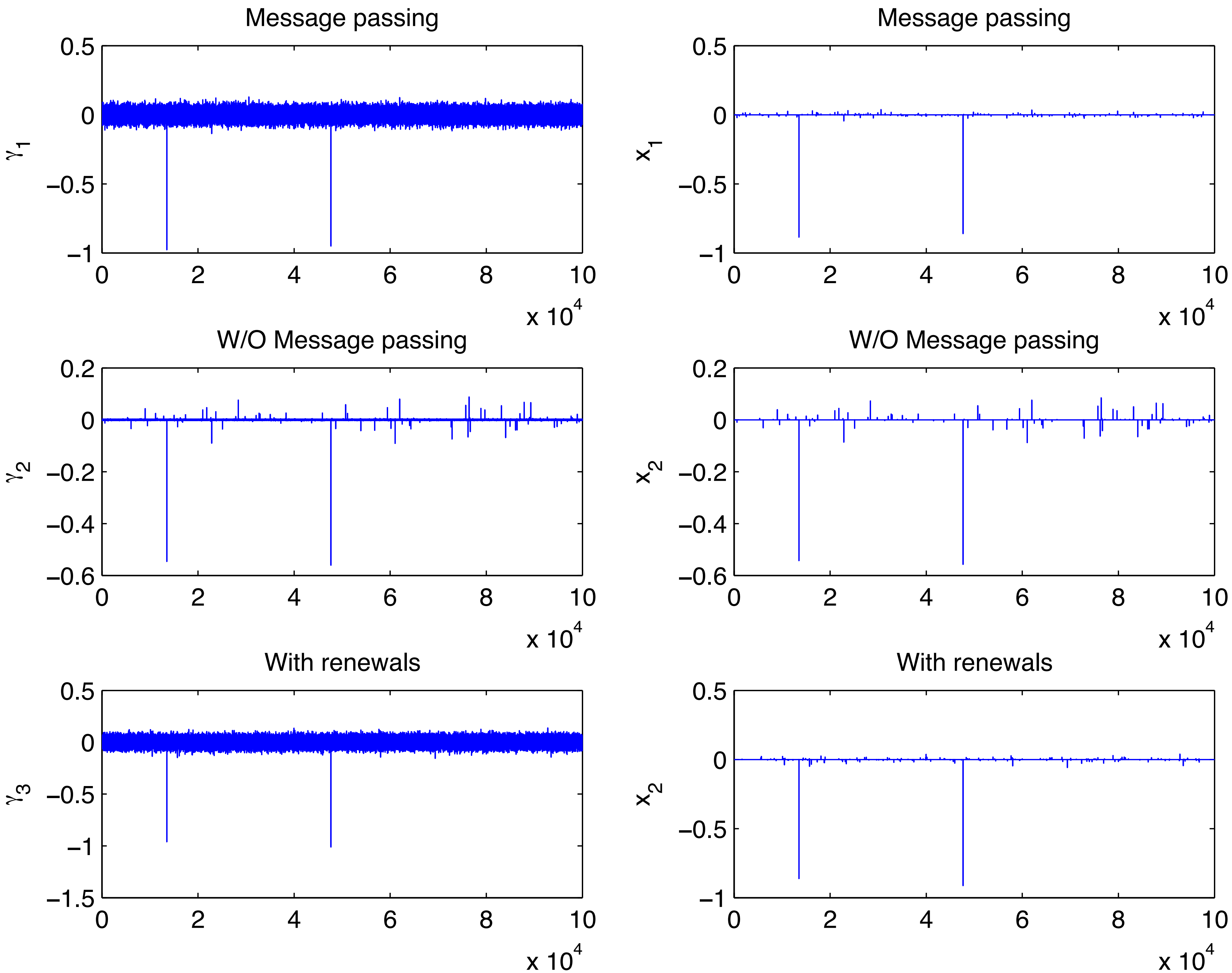
Message passing
Message passing
W/O Message passing
W/O Message passing
With renewals
With renewals
γ_1
γ_2
γ_3
x_1
x_2
x_2
x 10^4

Iteration 2



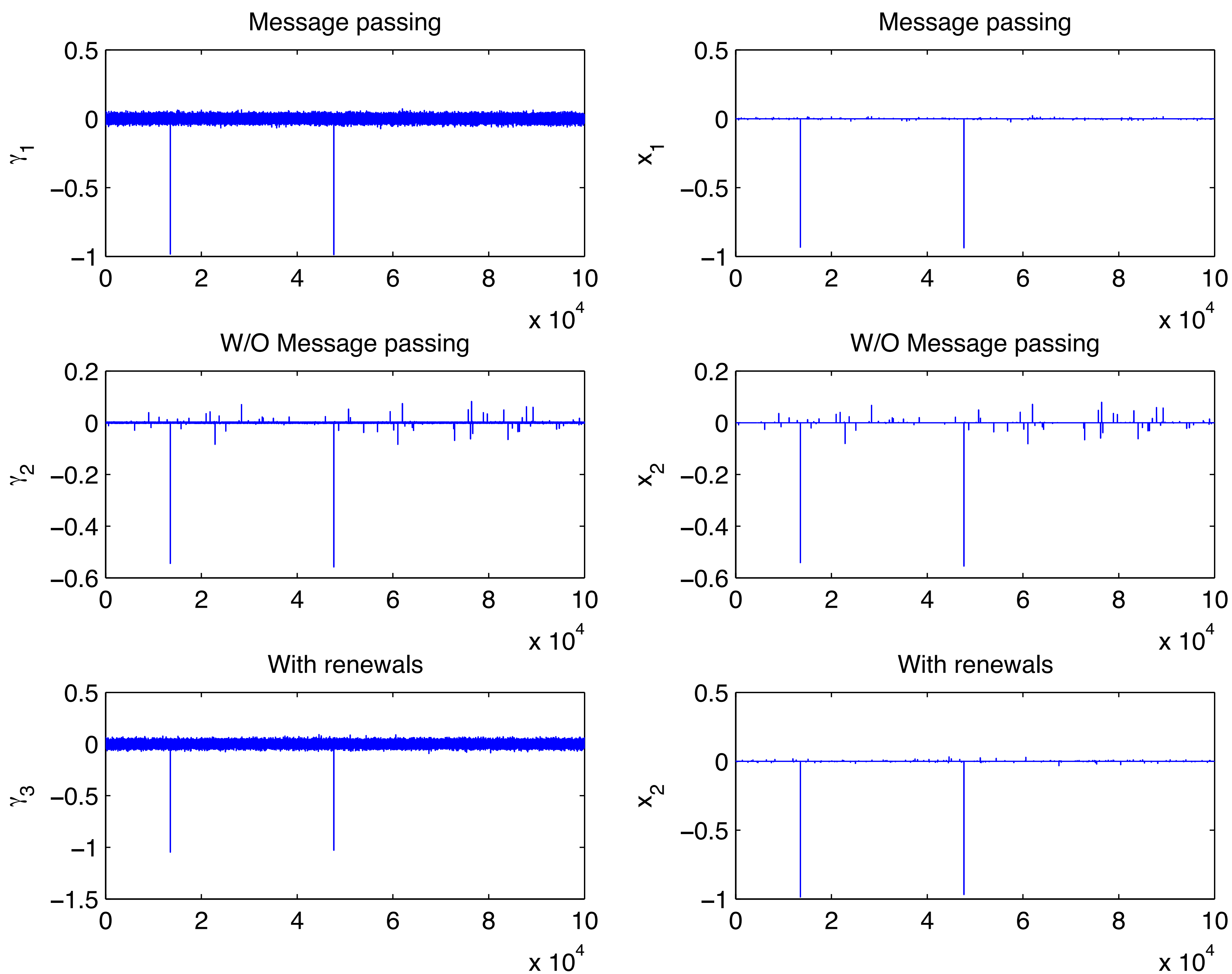
Message passing
Message passing
W/O Message passing
W/O Message passing
With renewals
With renewals
γ_1
x_1
γ_2
x_2
γ_3
x_2
x 10^4

Iteration 3



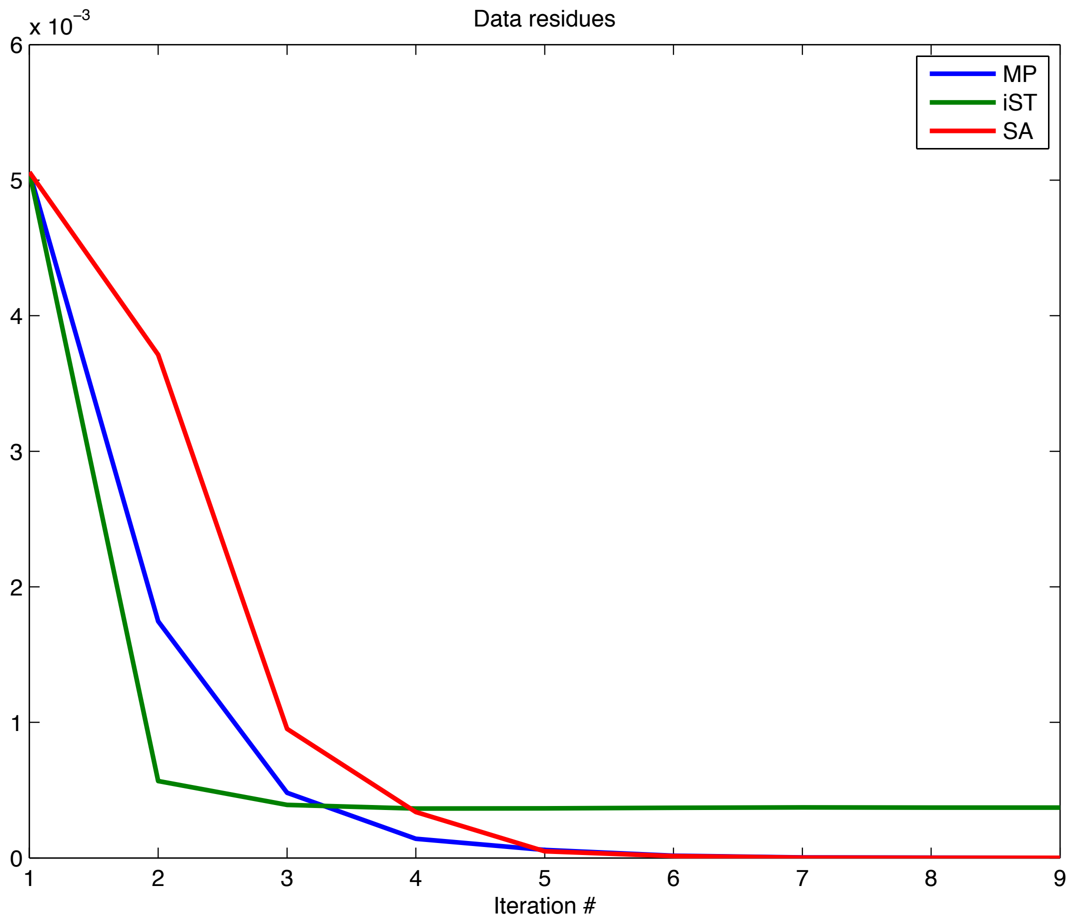
Message passing
γ_1
0.5
0
−0.5
−1
0 2 4 6 8 10
x 10^4
Message passing
x_1
0.5
0
−0.5
−1
0 2 4 6 8 10
x 10^4
W/O Message passing
γ_2
0.2
0
−0.2
−0.4
−0.6
0 2 4 6 8 10
x 10^4
W/O Message passing
x_2
0.2
0
−0.2
−0.4
−0.6
0 2 4 6 8 10
x 10^4
With renewals
γ_3
0.5
0
−0.5
−1
−1.5
0 2 4 6 8 10
x 10^4
With renewals
x_2
0.5
0
−0.5
−1
0 2 4 6 8 10
x 10^4

Iteration 4

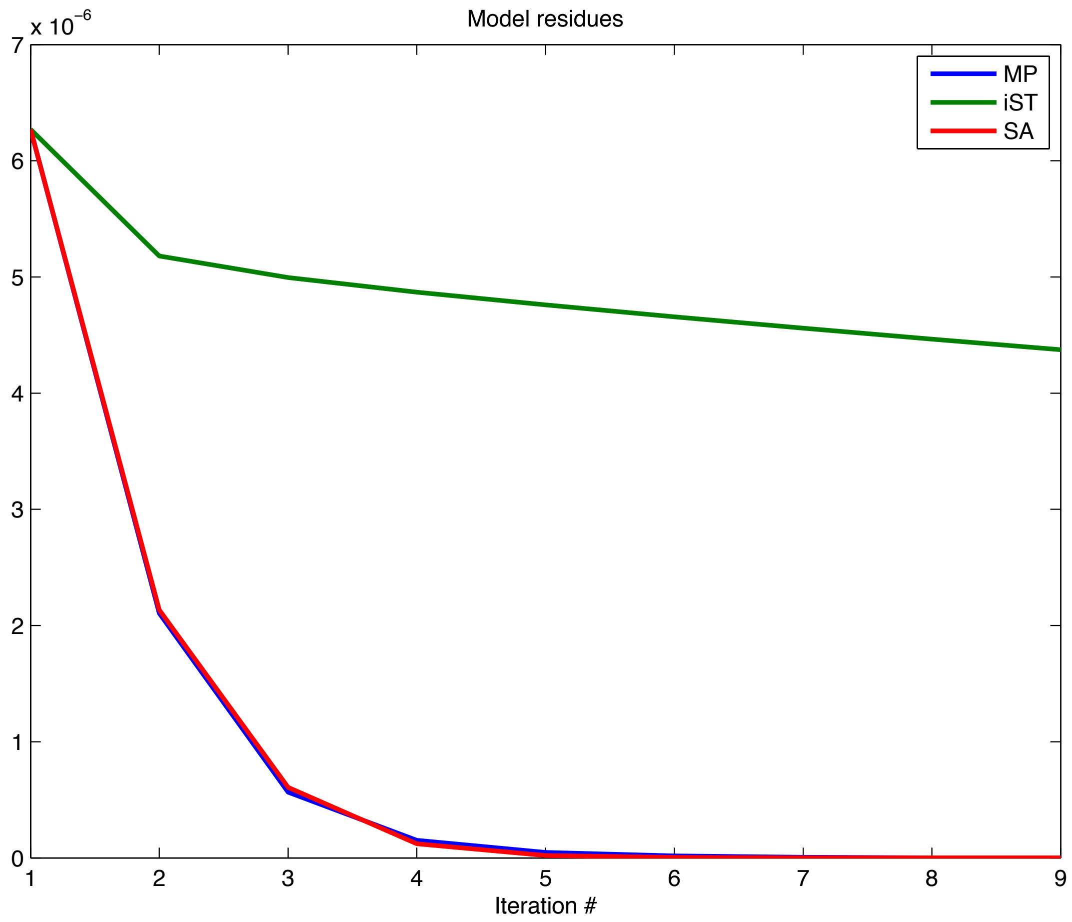

Message passing
0.5
0
−0.5
−1
γ_1
0
2
4
6
8
10
x 10^4
Message passing
0.5
0
−0.5
−1
x_1
0
2
4
6
8
10
x 10^4
W/O Message passing
0.2
0
−0.2
−0.4
−0.6
γ_2
0
2
4
6
8
10
x 10^4
W/O Message passing
0.2
0
−0.2
−0.4
−0.6
x_2
0
2
4
6
8
10
x 10^4
With renewals
0.5
0
−0.5
−1
−1.5
γ_3
0
2
4
6
8
10
x 10^4
With renewals
0.5
0
−0.5
−1
x_2
0
2
4
6
8
10
x 10^4

Residues

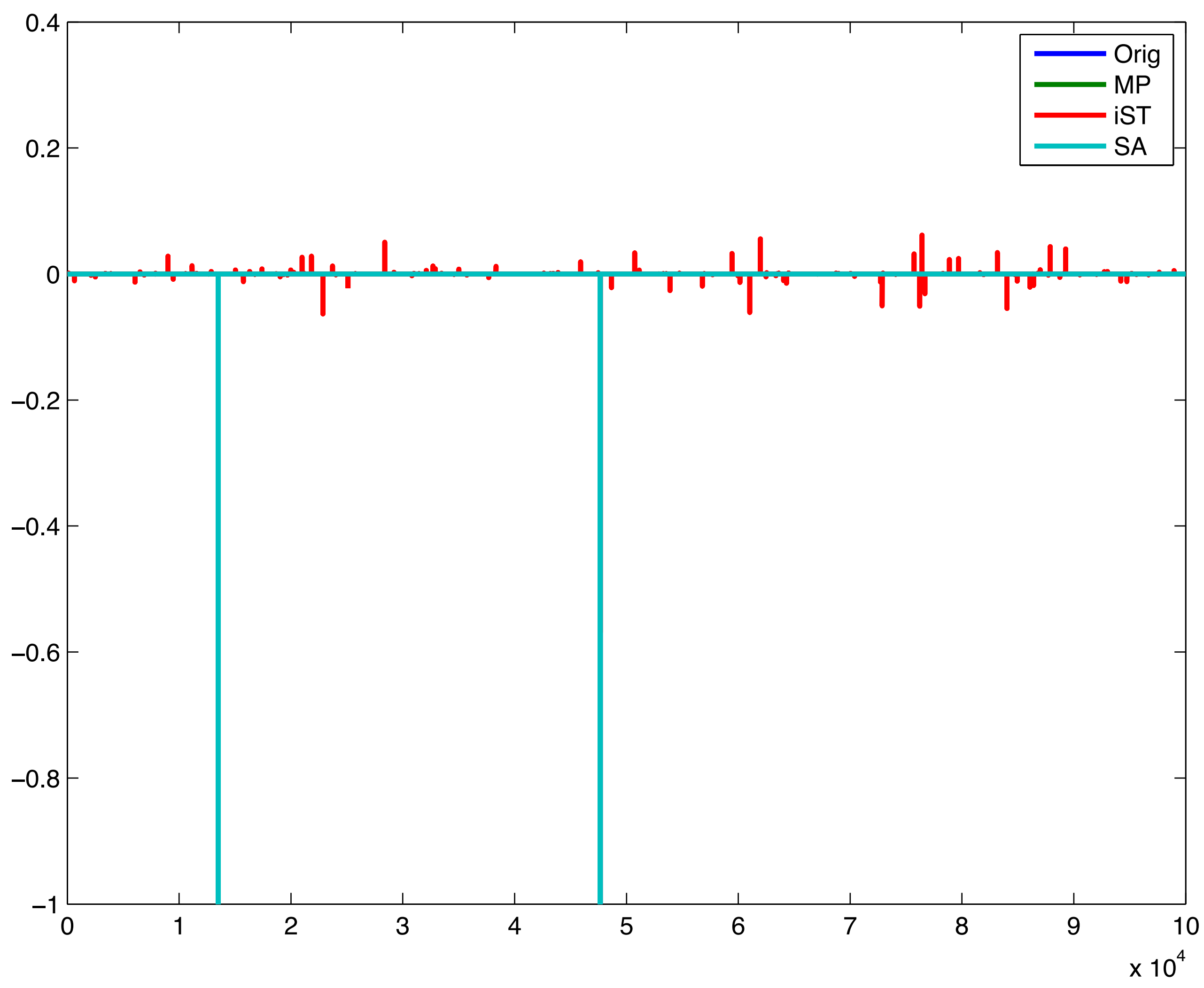
Data



Model



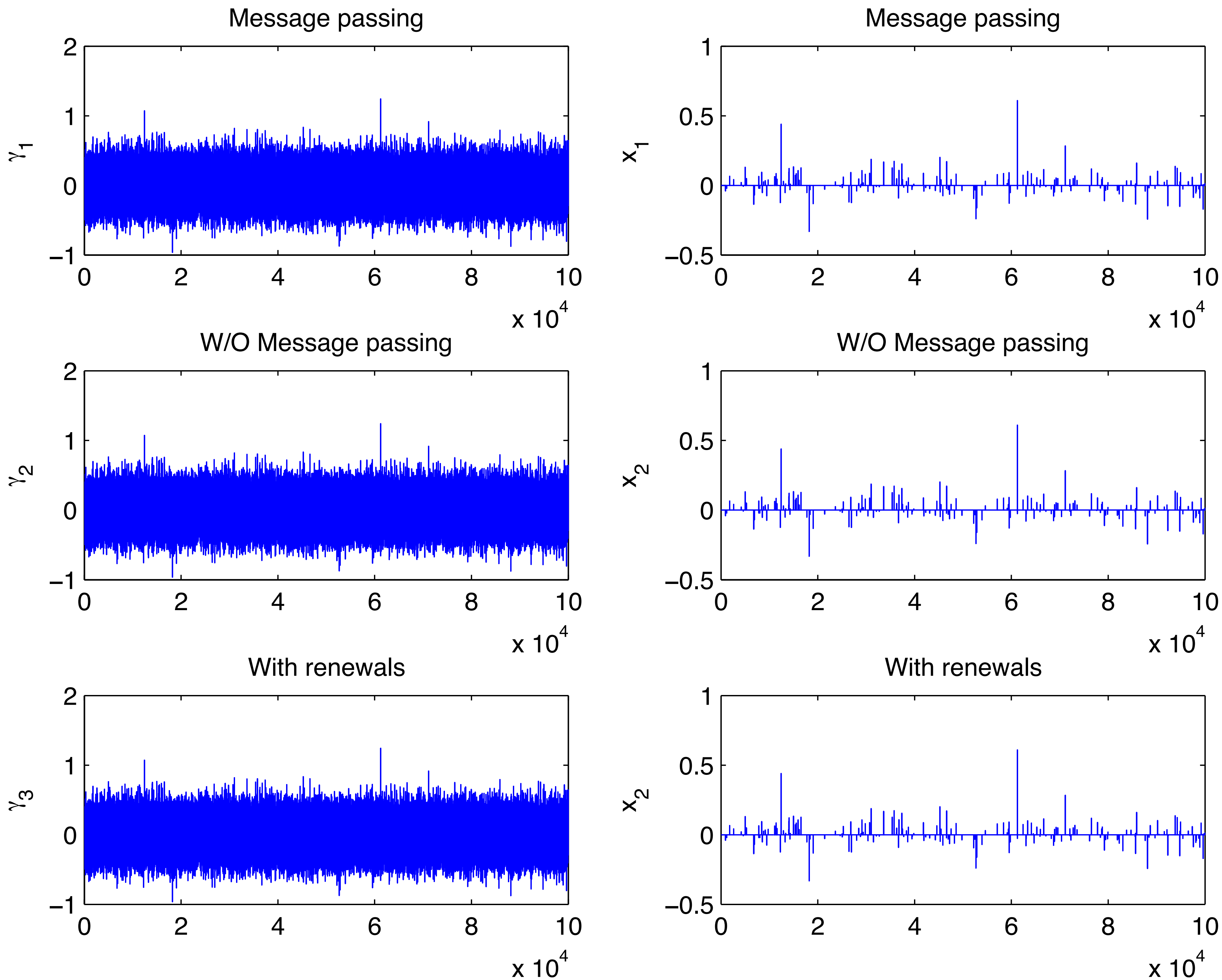
Recovery



Setup

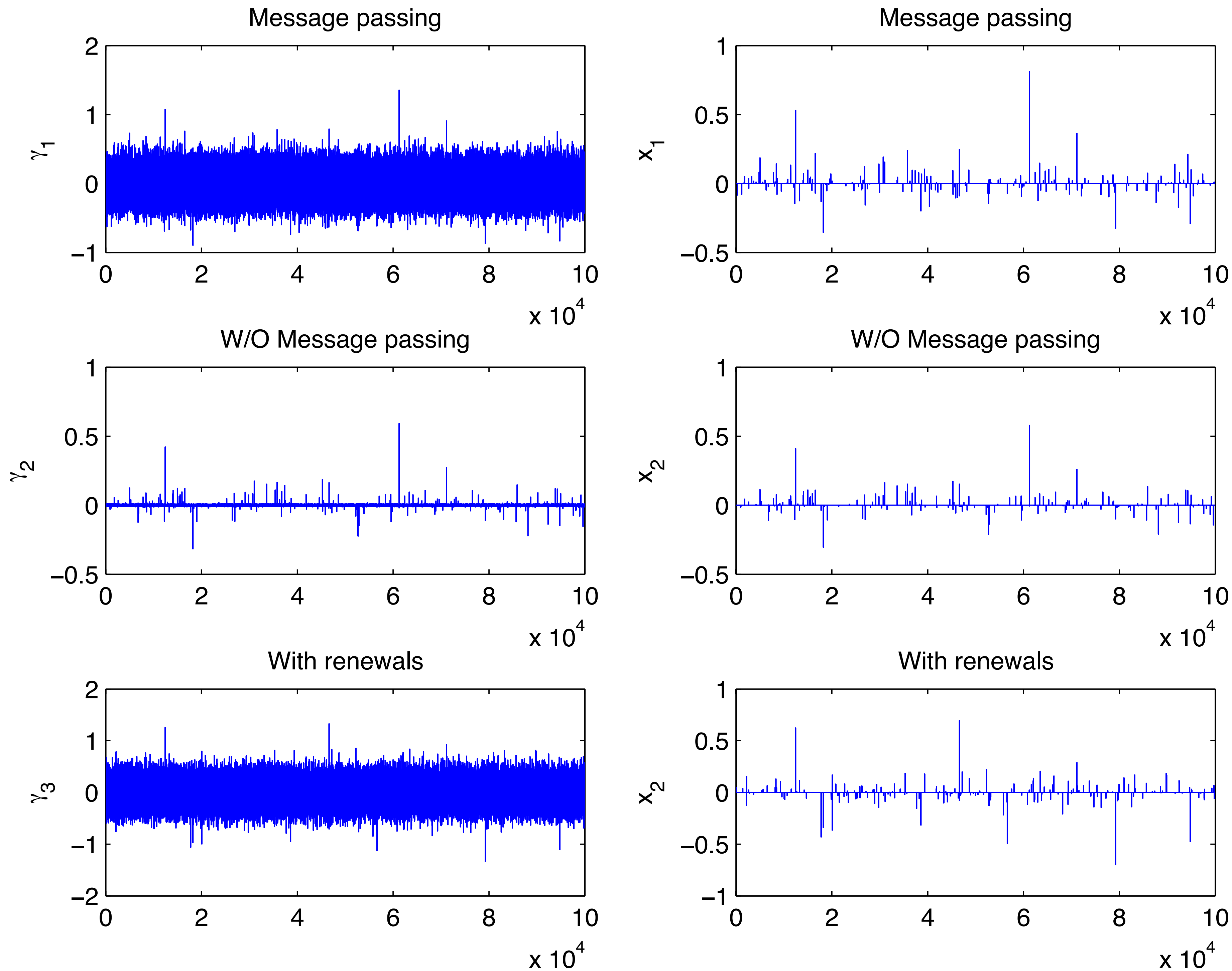
```
% Number of iterations of the algorithm
T = 200;

% Stopping criterion (tolerance for successfull decoding)
tol = 1e-4;
n = 200;
k = 10;
N = 100000;

A = (1/sqrt(n)) .* randn(n, N);

% Sparse signal (with uniform distribution of non-zeros)
x = [sign(rand(k,1) - 0.5); zeros(N-k,1)];
x = x(randperm(N));

% Generate Measurements
b = A*x;
```

Iteration 1



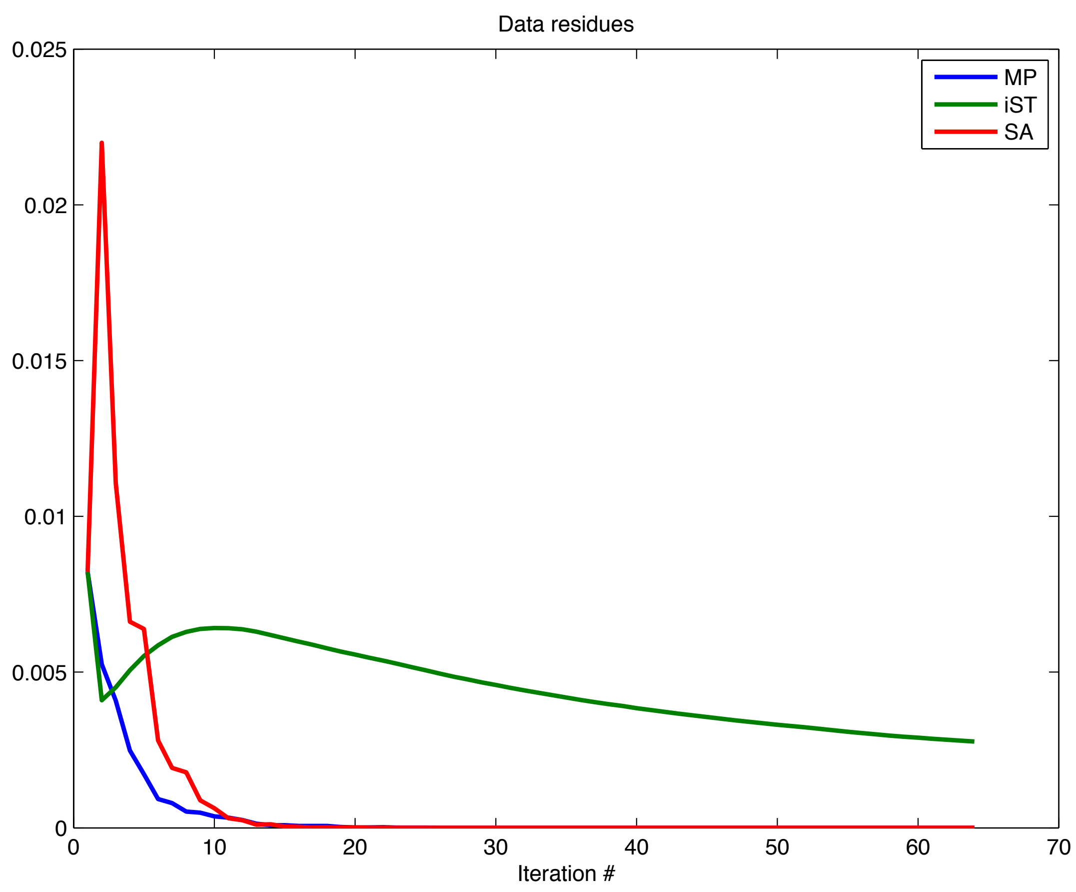
Message passing
W/O Message passing
With renewals
γ_1
γ_2
γ_3
x_1
x_2
x_2
x 10^4

Iteration 2

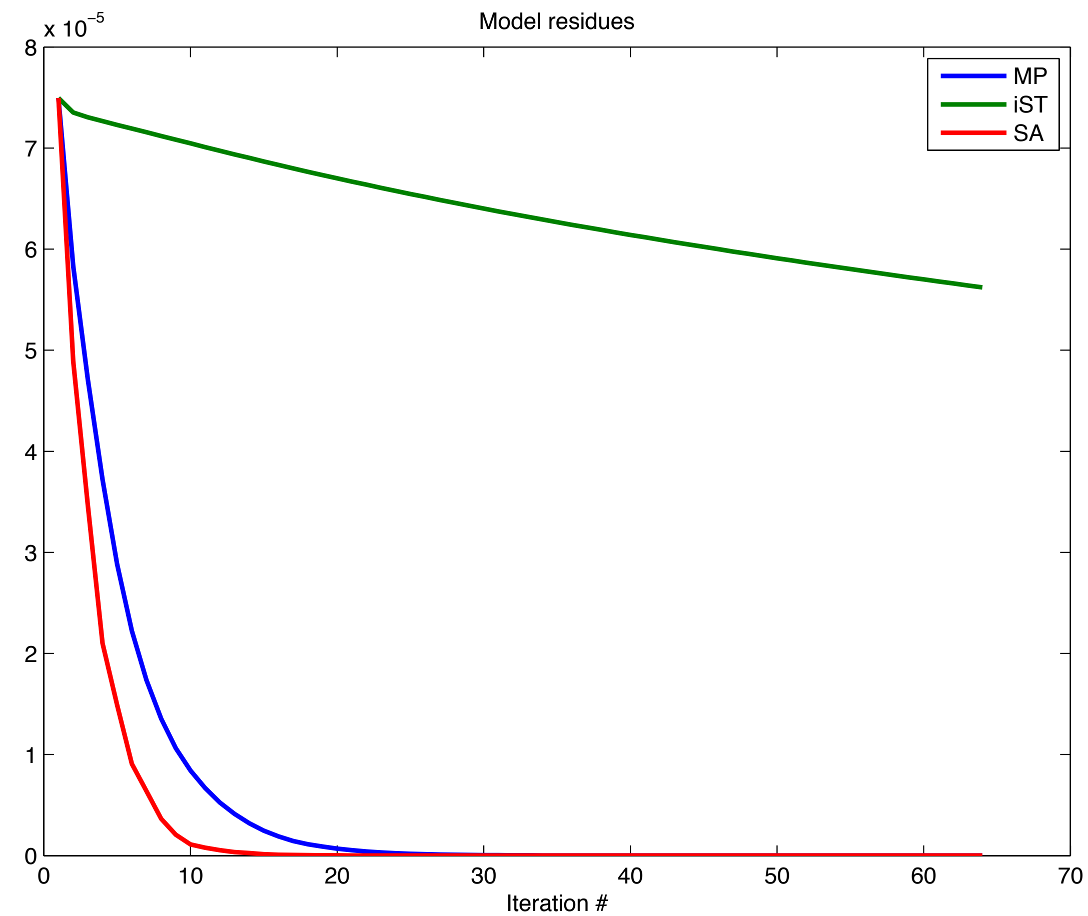

Message passing
γ1
x1
W/O Message passing
γ2
x2
With renewals
γ3
x2
x 10^4

Residues

Data

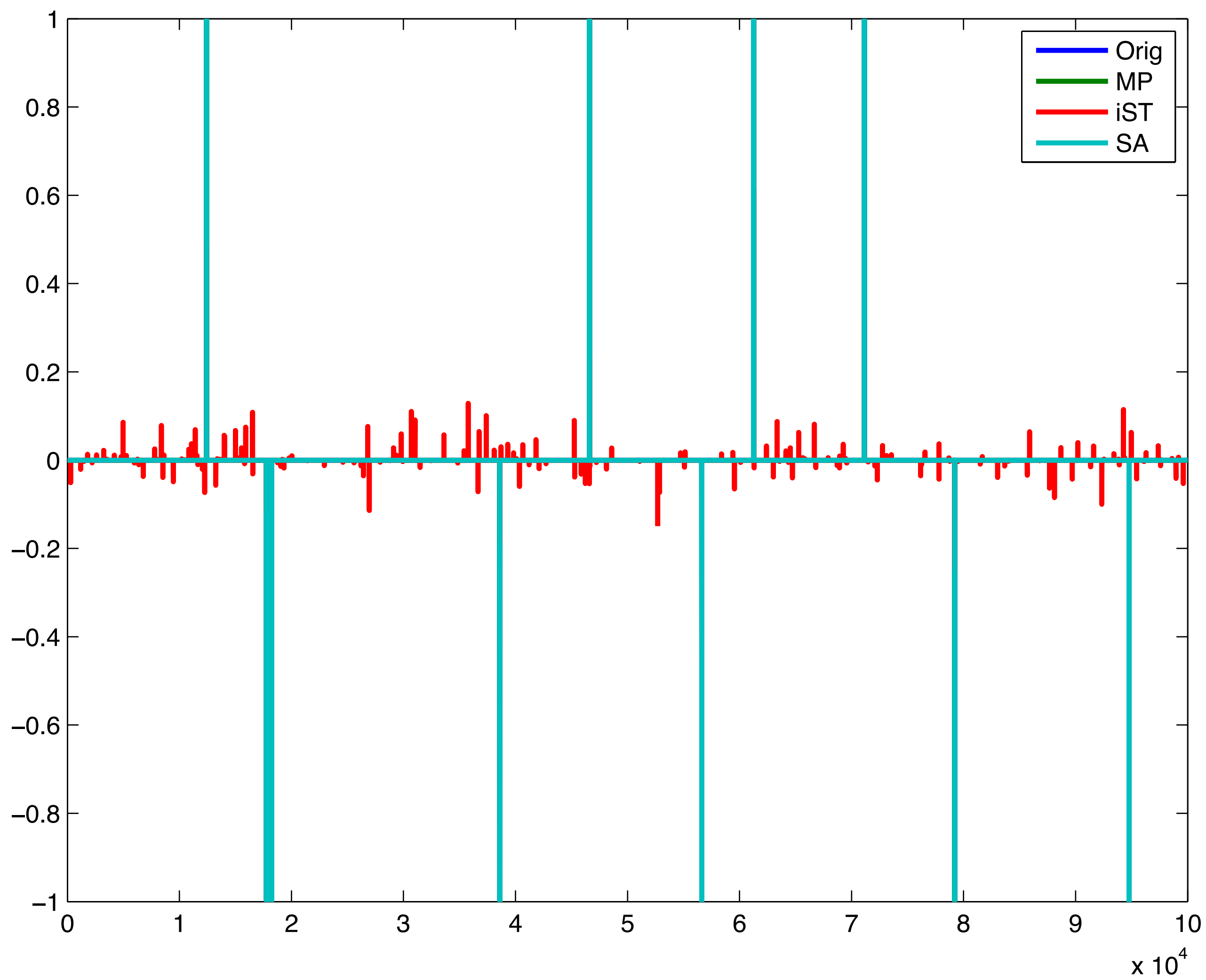


Model



SLIM

Recovery



Observations

Approximate message passing:

- is *connected* to the *stochastic approximation* because it *draws* a *new* matrix and *data* for *each* iteration
- *differs* from *stochastic* gradients because it *relies* on
  - a *nonlinearity* in the form of *tuned* thresholding
  - very *particular* (Gaussian) matrices and *sparse* vectors

Recent proofs that BP is solved in the *large* scale *limit.*

Renewals (or message) are *responsible* for a *remarkable* speed up.

Conclusions

Emergence of 'batching ideas' for large-scale problems for which

- people chip away with small *randomized* subproblems
- *optimization* problems exist with *rigorous* convergence *proofs* but for which *convergence* is *rarely* attained in *practice*
- fast AMP *algorithms* exist that turn *iterative* soft thresholding into *iterative* denoising, which in the large-scale limit correspond to solving BP

For the second category, *extreme* size & *complexity* of our problems may actually *work* to our *advantage*...