Recent progress of bidirectional deconvolution

SEP-147 p333

Qiang Fu

Stanford Exploration Project

Bidirectional convolution

- Convolution model

$$d = r * w$$

Where d = data, w = wavelet,

Bidirectional convolution

- Convolution model

$$d = r * w$$

- Bidirectional convolution

$$d = r * w$$

$$d = r * (w_a * w_b^r)$$

Where d = data, $w_a * w_b^r = w$ = wavelet,

w_a and w_b are both minium phase

(The superscript r means reverse in time)

Bidirectional deconvolution

- Convolution model

$$d = r * w$$

- Bidirectional deconvolution

$$d = r * w$$

$$d = r * (w_a * w_b^r)$$

$$r = d * (w_a * w_b^r)^{-1}$$

Where d = data, $w_a * w_b^r = w$ = wavelet,

w_a and w_b are both minium phase

(The superscript r means reverse in time)

Bidirectional deconvolution

- The deconvolution filters are the inverse wavelets

$$\begin{cases} w_a * f_a = \delta \\ w_b * f_b = \delta \end{cases}$$

Bidirectional deconvolution

- The deconvolution filters are the inverse wavelets

$$\begin{cases} w_a * f_a = \delta \\ w_b * f_b = \delta \end{cases}$$

- Apply deconvolution filters on the data to recover the reflectivity series

$$r = d * f_a * f_b^r$$

Bidirectional deconvolution

- The deconvolution filters are the inverse wavelets

$$\begin{cases} w_a * f_a = \delta \\ w_b * f_b = \delta \end{cases}$$

- Apply deconvolution filters on the data to recover the reflectivity series

$$r = d * f_a * f_b^r$$

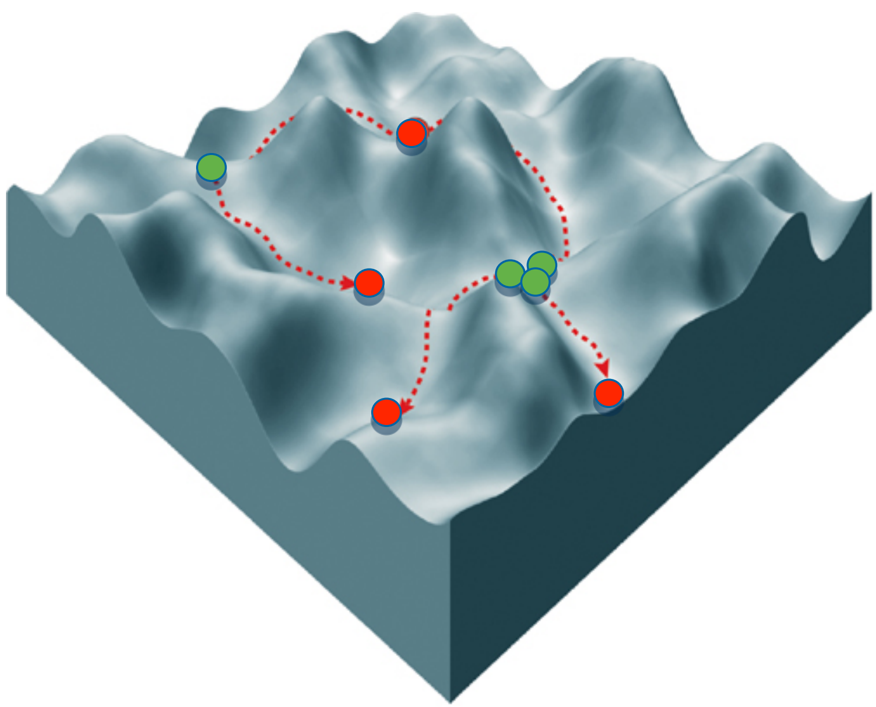
- This deals with mix-phase wavelets

Time-domain methods

- Two time-domain methods
  - Slalom method
    - Zhang, Y. and J. Claerbout, 2010, A new bidirectional deconvolution method that overcomes the minimum phase assumption: SEP-Report, 142, 93–104.
  - Symmetric method
    - Shen, Y., Q. Fu, and J. Claerbout, 2011, A new algorithm for bidirectional deconvolution: SEP-Report, 143, 271–282.
- However, both time methods are very sensitive to the starting solution and the parameters
  - For example, changing the filter length a little led to total different result

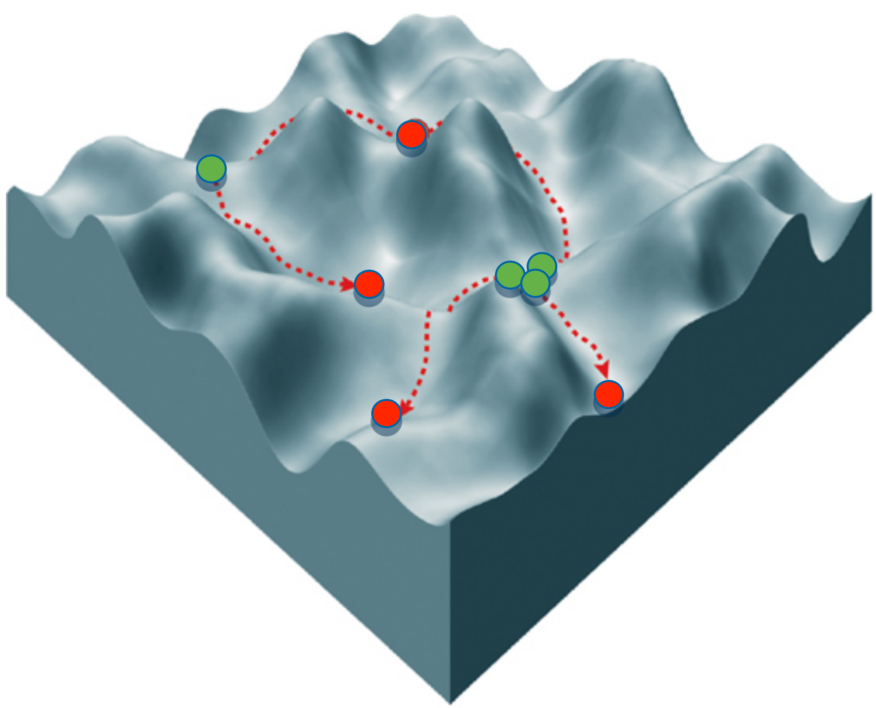
Instability caused by nonlinearity?

- Bidirectional deconvolution is a non-linear problem
  - A multidimensional non-linear objective function with local minima



Instability caused by nonlinearity?

- Bidirectional deconvolution is a non-linear problem
  - A multidimensional non-linear objective function with local minima
  - Results are very sensitive to the starting solution and parameters



Instability caused by null space?

- The sensitivity of initial solution may be caused by the Null Space
  - We do not have enough evidence to confirm the reason yet
  - No matter what is the reason of this sensitivity, we need to solve this problem by preconditioning in time-domain methods

Preconditioning

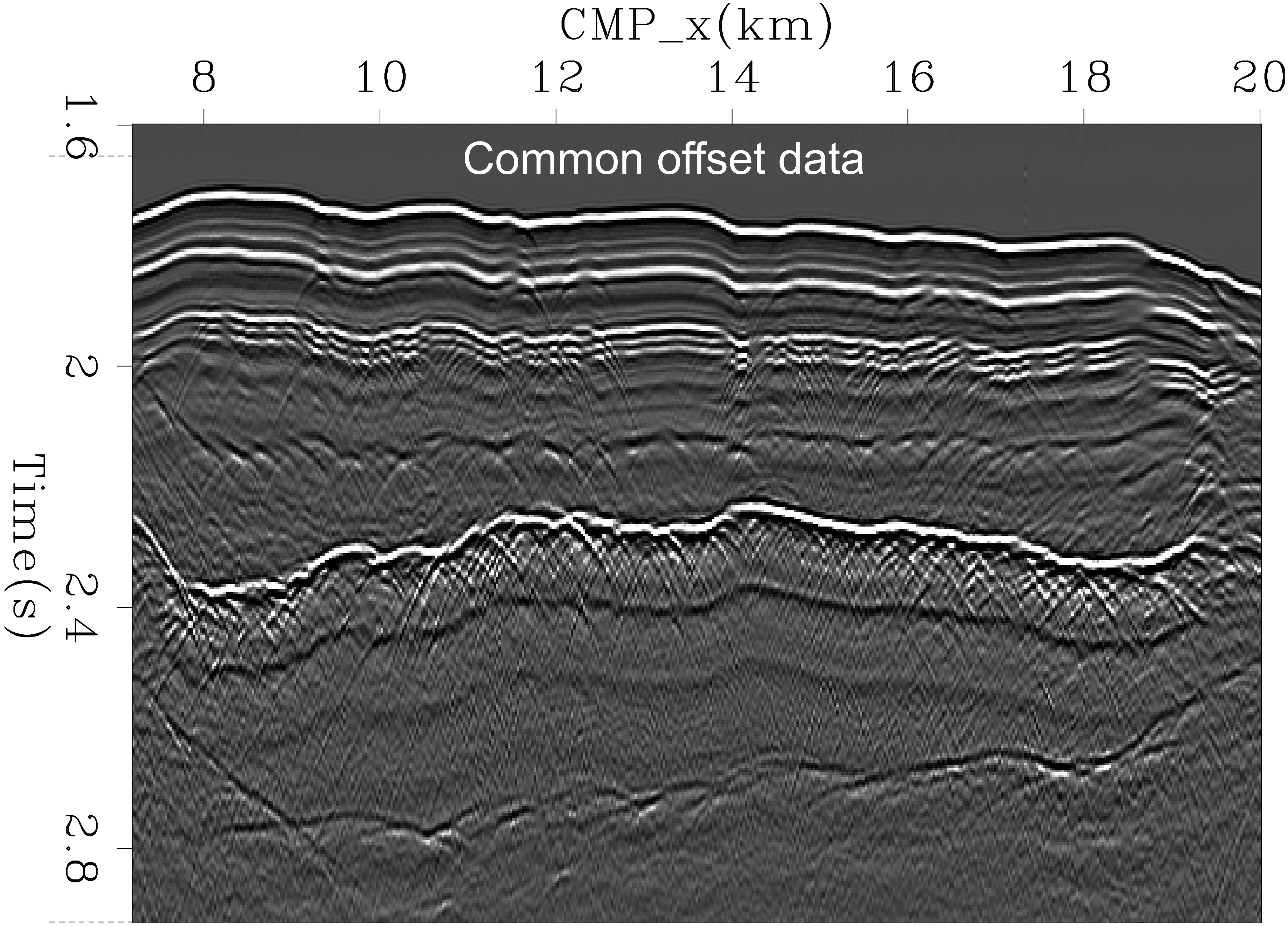
- Time domain methods require preconditioning to provide prior information in inversion
  - Also accelerates the convergence and stabilizes the results

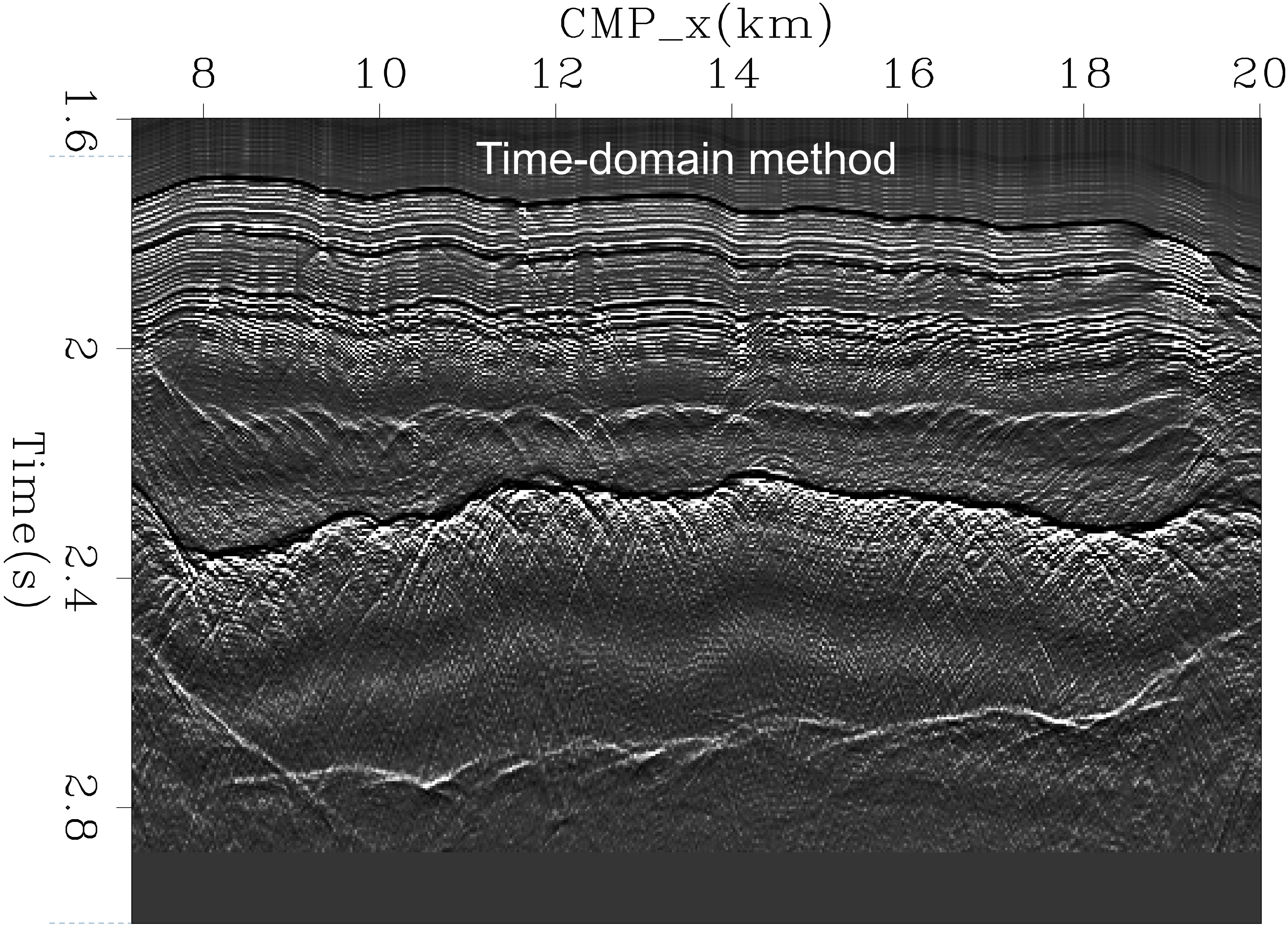
Preconditioning

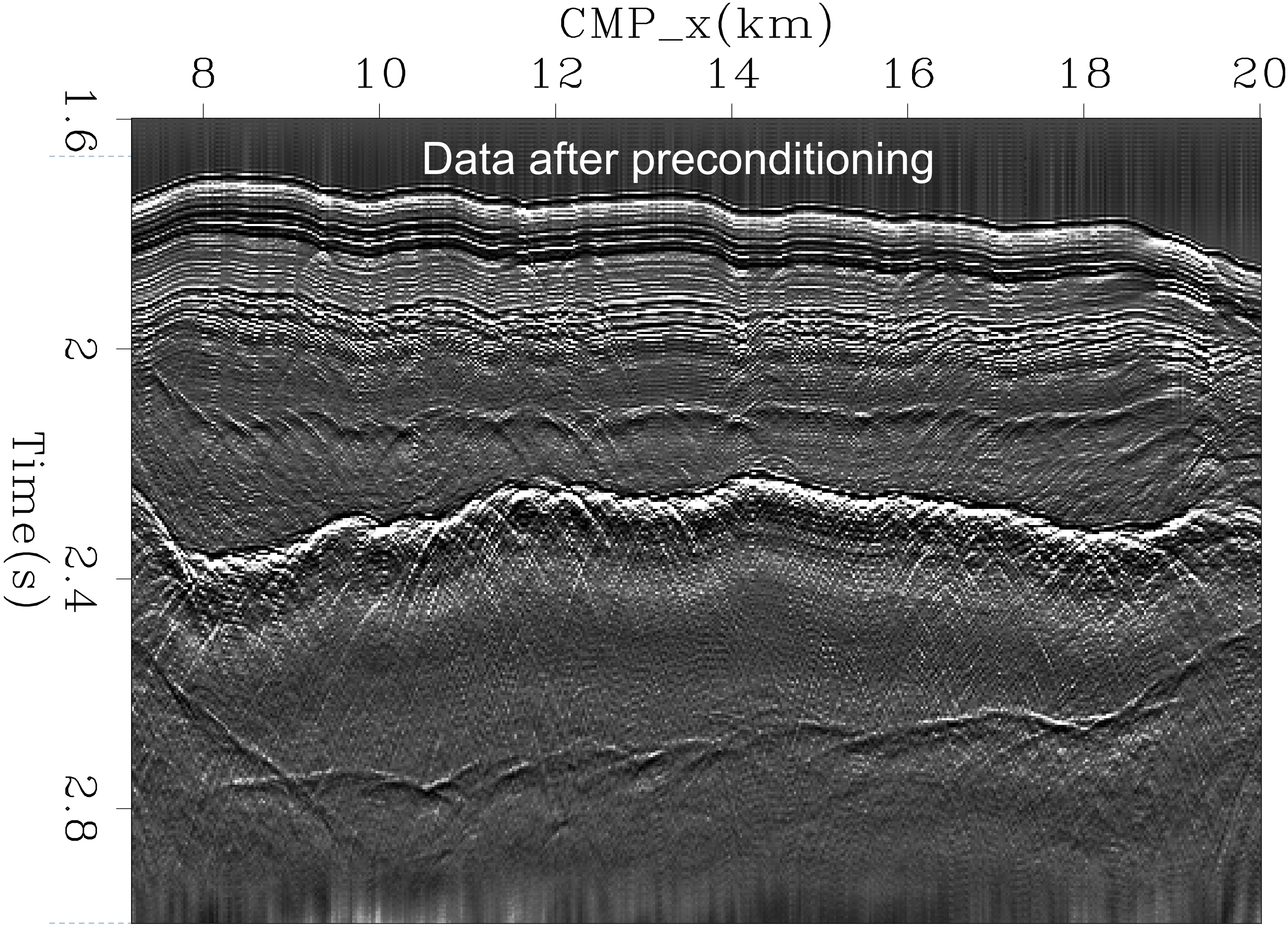
- We use PEF (prediction error filter) as a preconditioner for time domain methods

Preconditioning

- We use PEF (prediction error filter) as a preconditioner for time domain methods
- PEF is a causal and minimum-phase filter
  - That means if we have a Ricker wavelet in our data, we can only get an output spike on the first lobe

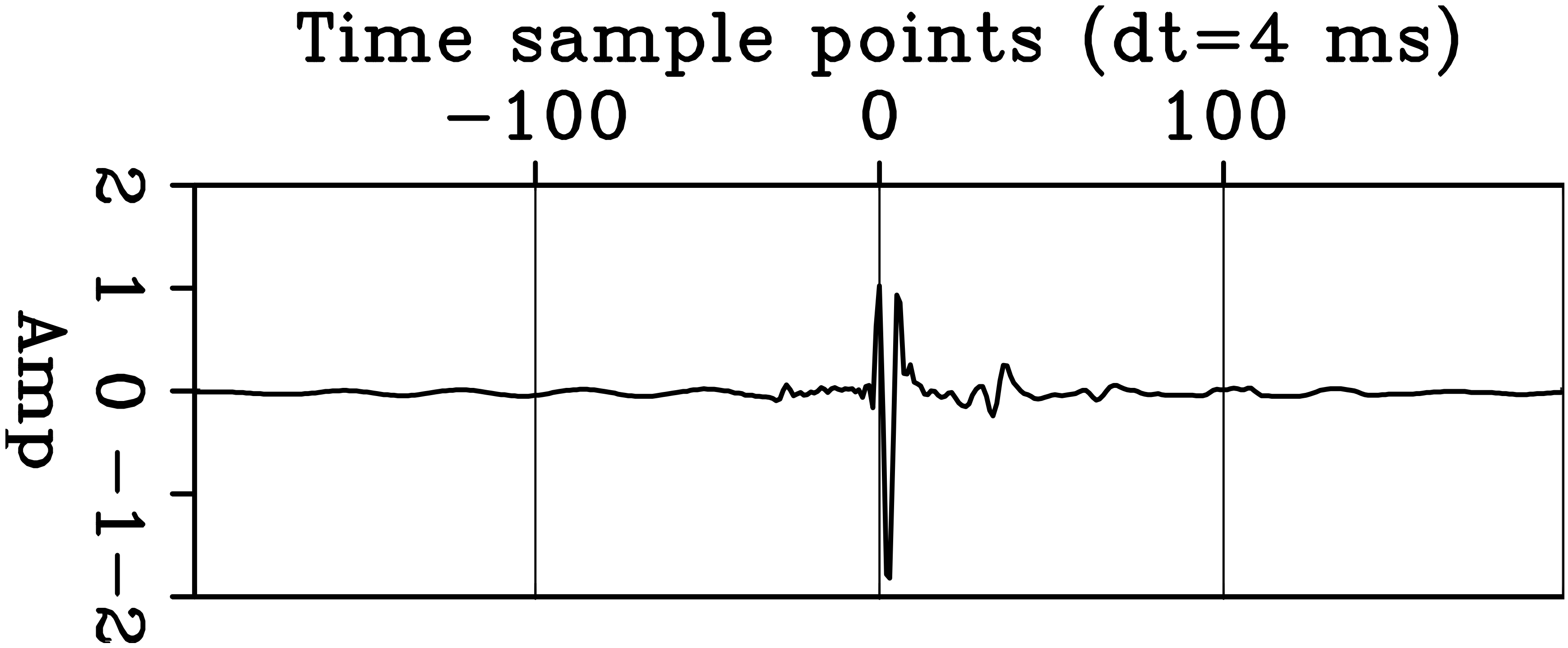

CMP_x(km)
8
10
12
14
16
18
20
1.6
2
2.4
2.8
Time(s)
Common offset data


CMP_x(km)
8
10
12
14
16
18
20
Time-domain method
Time(s)
1.6
2
2.4
2.8



CMP_x(km)
8
10
12
14
16
18
20
1.6
2
2.4
2.8
Time(s)
Data after preconditioning

Estimated wavelet from time-domain method



The time-domain methods - issue

- Polarity flipping and time shifts:
  - The polarities of the events are flipped by bidirectional deconvolution (white to black)
  - There is a time shift for the peaks of the events after bidirectional deconvolution
- This reminds us the preconditioner may lead to problems
  - We don't want to rely on the preconditioner

Logarithm method

- Redefine the unknowns *U*

$$r = d * f_a * f_b = \mathrm{IFT}(DF_aF_b) = \mathrm{IFT}(DF)$$

$$U = \log F = \log F_aF_b$$

$$u = \mathrm{IFT}(U)$$

- The final deconvolution filter is

$$f = f_a * f_b = \mathrm{IFT}(e^U)$$

Logarithm method

- Why do we need this?
  - We have to update minimum-phase and maximum-phase deconvolution filters respectively
  - The exponential "e" helps to map minimum-phase and maximum-phase filters into *u* separately without crosstalk

$$u = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0, u_1, u_2, u_3, \ldots)$$

Logarithm method

- Why do we need this?
  - We have to update minimum-phase and maximum-phase deconvolution filters respectively
  - The exponential "e" helps to map minimum-phase and maximum-phase filters into *u* separately without crosstalk

$$u = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0, u_1, u_2, u_3, \ldots)$$

f_b f_a

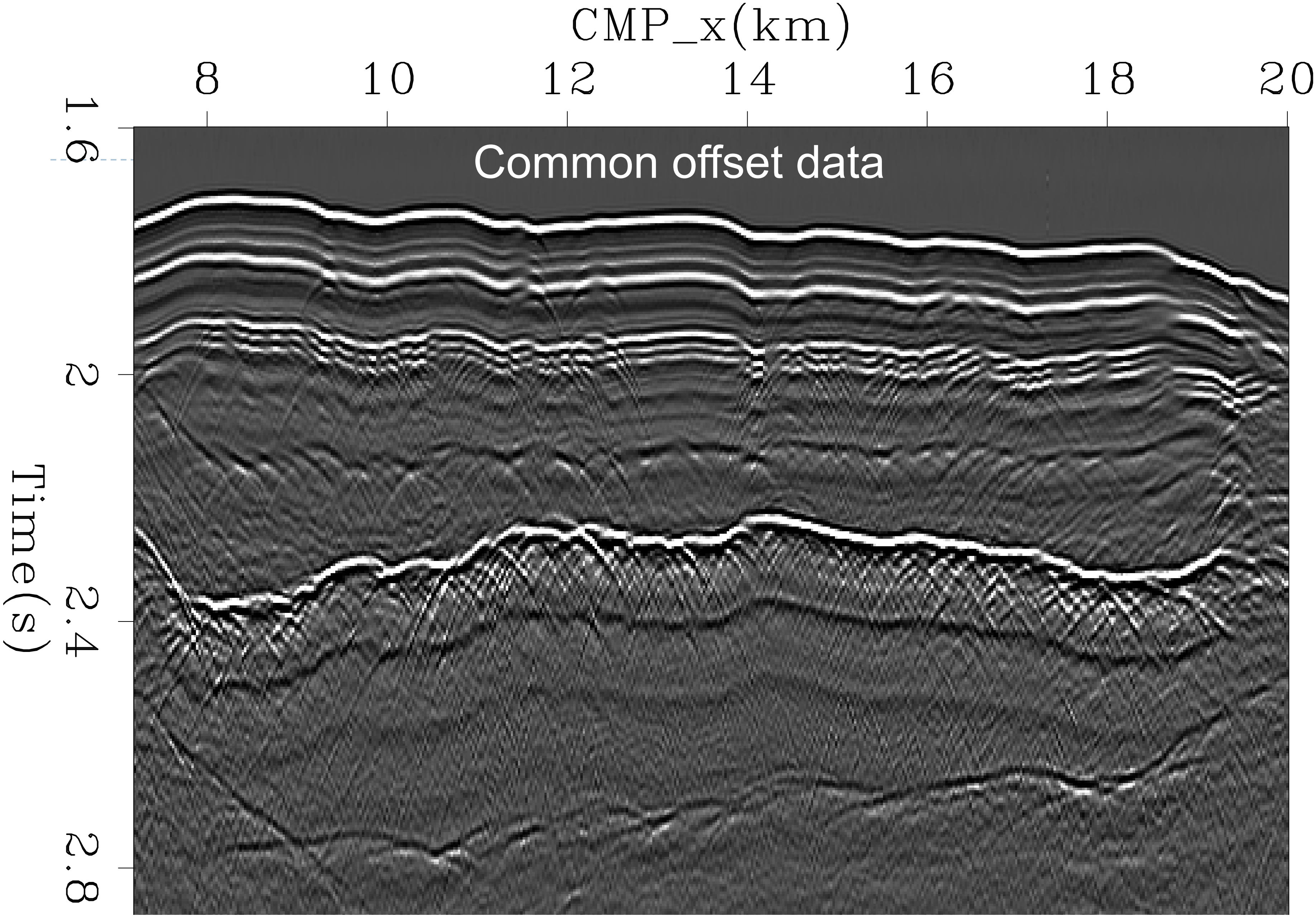
- Claerbout, J., Q. Fu, and Y. Shen, 2011, A log spectral approach to bidirectional deconvolution: SEP-Report, 143, 297–300.

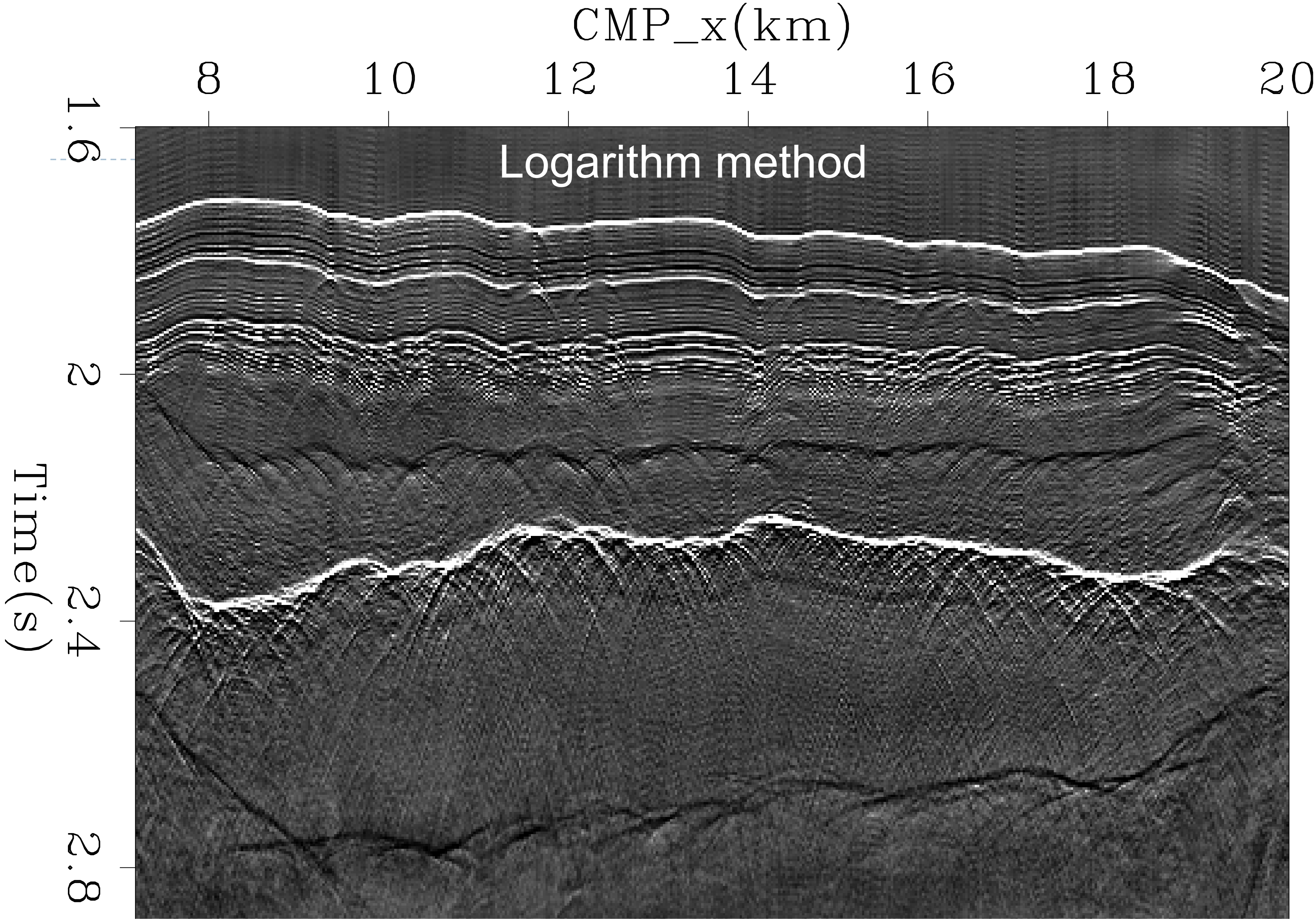
Logarithm method is self-preconditioned

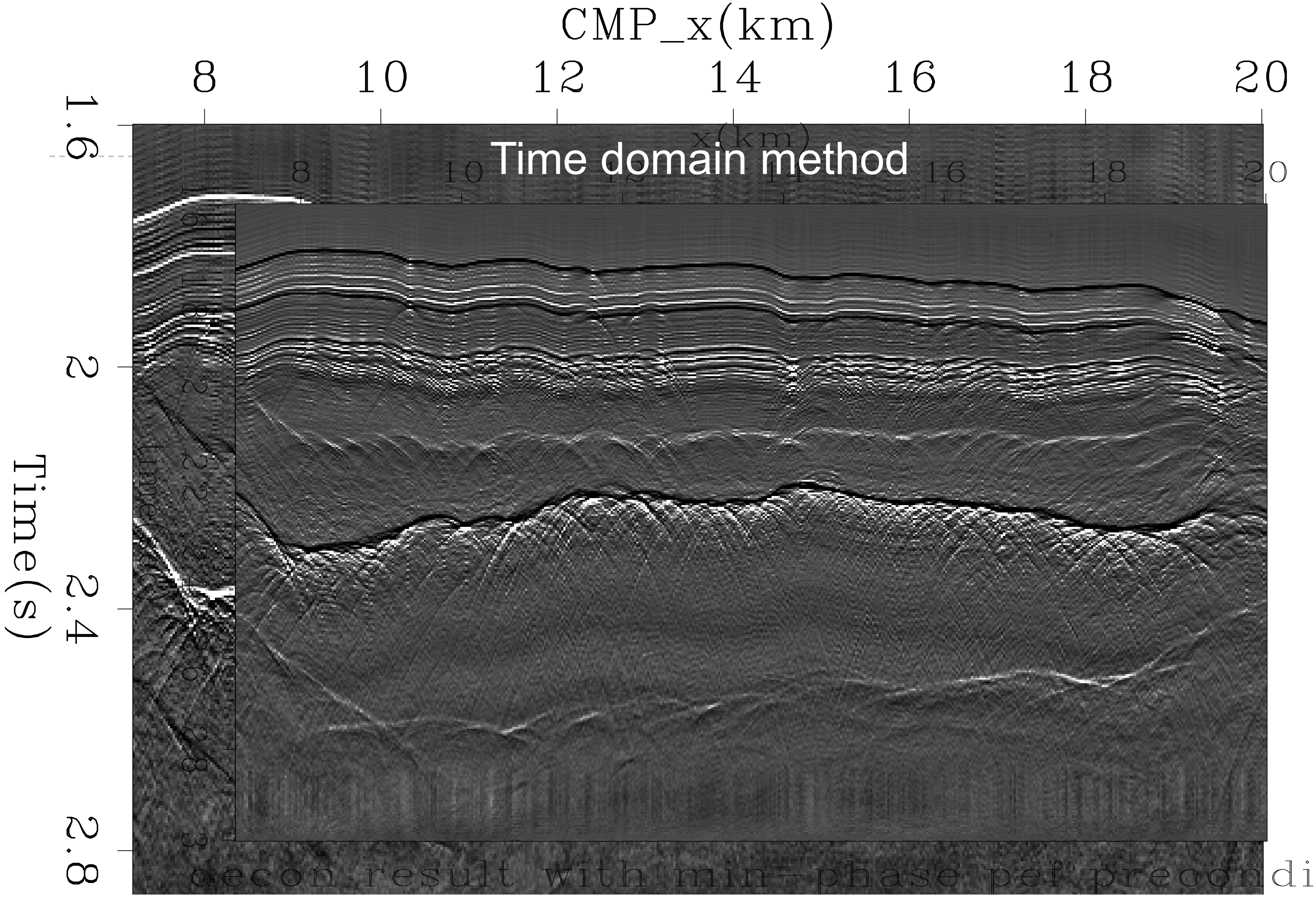
- The logarithm method is self-preconditioned

$$r = \mathrm{IFT}(De^{U+\alpha\Delta U}) = \mathrm{IFT}(De^{U}e^{\alpha\Delta U}) = d * \boxed{f_{pre}} * f_{update}$$

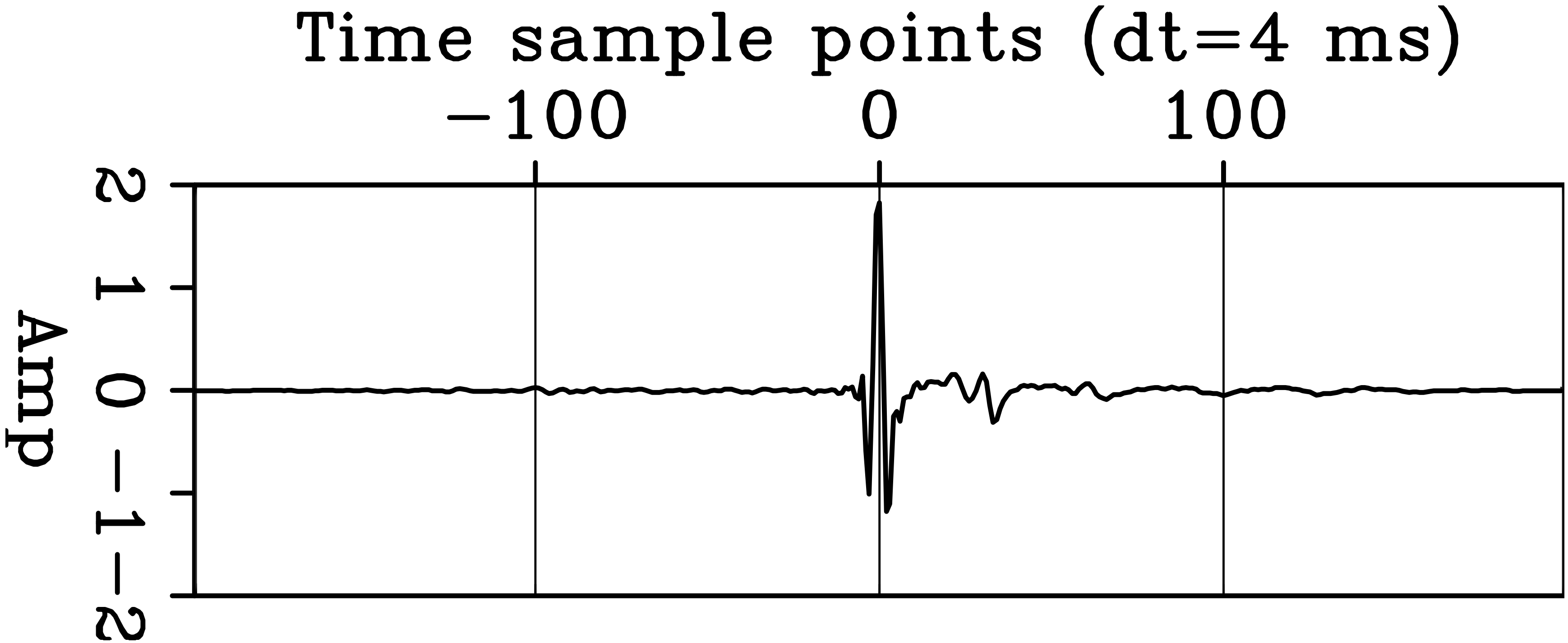
- We do not need extra preconditioning for this method anymore
- The convergence speed is fast
- We do not have any polarity flips or time shifts in the logarithm method


CMP_x(km)
8
10
12
14
16
18
20
Common offset data
Time(s)
1.6
2
2.4
2.8

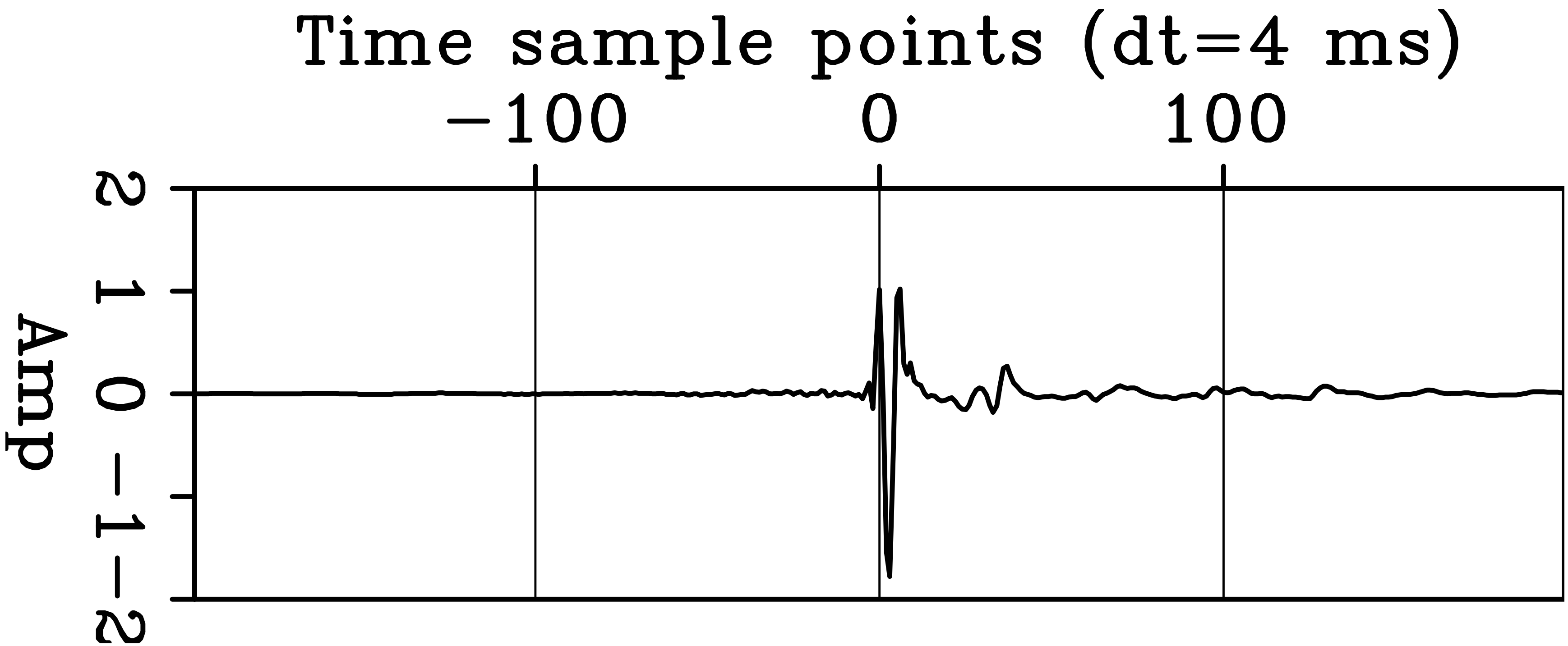

CMP_x(km)
8
10
12
14
16
18
20
Logarithm method
Time(s)
1.6
2
2.4
2.8


CMP_x(km)
8
10
12
14
16
18
20
Time domain method
x(km)
Time(s)
1.6
2
2.4
2.8
decon result with min-phase pef precondi

Estimated wavelet: logarithm method



Estimated wavelet: time-domain method



Conclusion - advantages

- The logarithm method is self-preconditioned. We do not have any polarity flips or time shifts in the logarithm method and the convergence is fast
- The f_a and f_b are guaranteed to be minimum-phase and maximum-phase respectively (the time domain methods can not guarantee this)
- We have only one rather than two-coefficient series to solve for

Conclusion - disadvantages

- We need to constrain the deconvolution filter
  - The estimated deconvolution filter is as long as the input data trace
- However, it is not easy.
  - because of the exponential, the **u** is not linear with the deconvolution filter **f**. Thus it is not easy to constrain the filter length

Acknowledgements

- I would especially thank
  - Jon Claerbout

  for all knowledge he teaches me
- I would like to thank
  - Yang Zhang
  - Antoine Guitton
  - Shuki Ronen

  for help on my research

THANK YOU

Backup slices

Hyperbolic penalty function

- We replace the conventional L2 norm with a hyperbolic penalty function.
  - This favors sparseness of the result after bidirectional deconvolution and retrieves non-white reflectivity series.

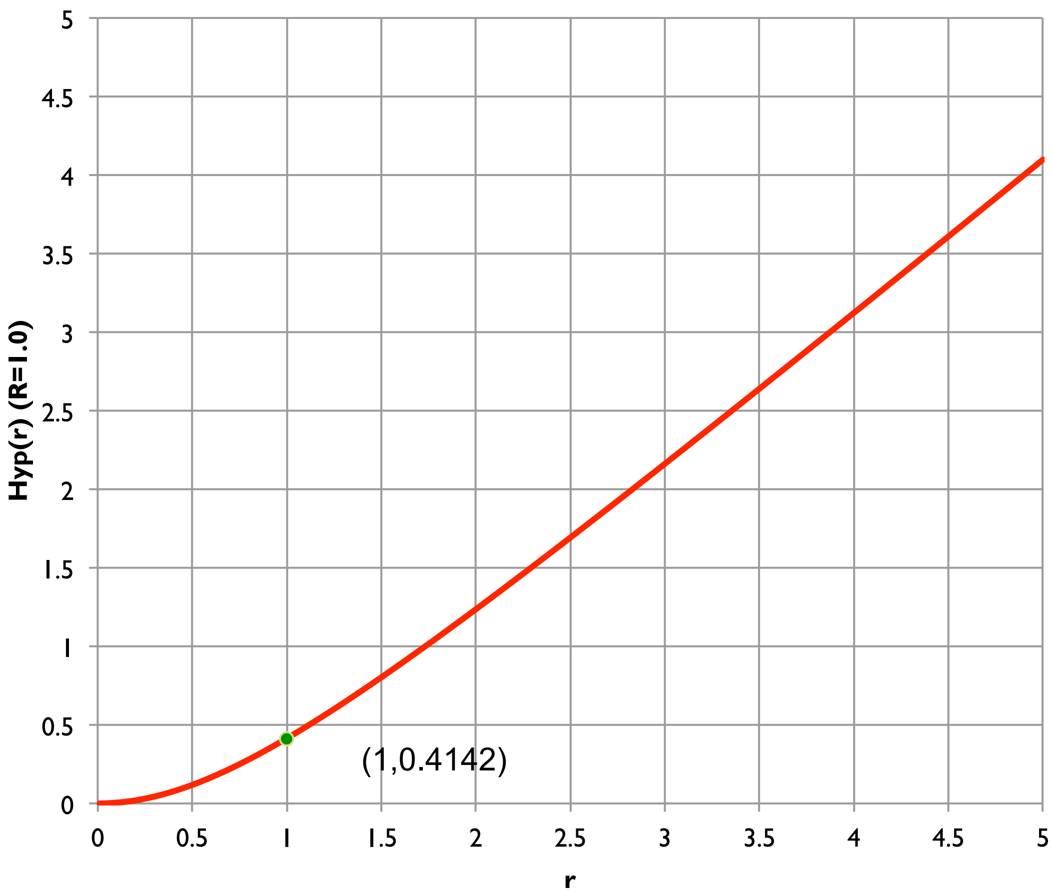
Hyperbolic penalty function

- We replace the conventional L2 norm with a hyperbolic penalty function.
  - This favors sparseness of the result after bidirectional deconvolution and retrieves non-white reflectivity series.
- Hyperbolic penalty function:

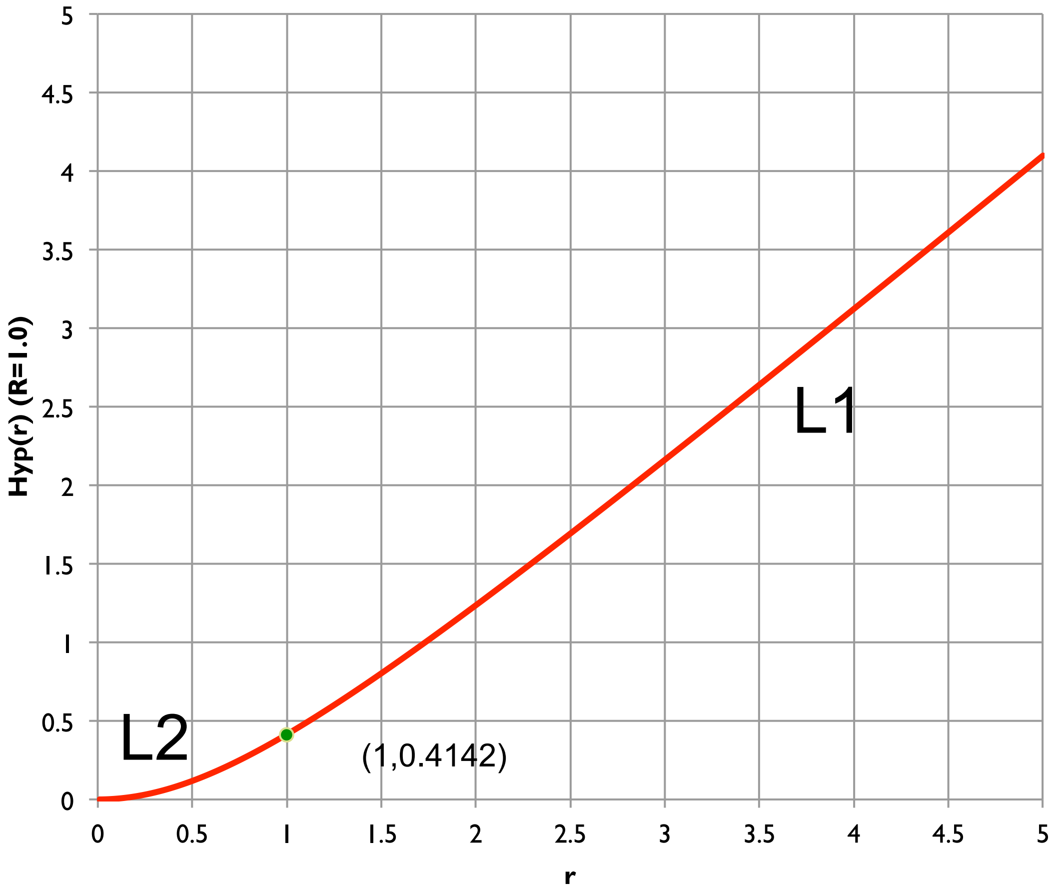
$$\mathrm{Hyp}(r) = \sqrt{r^2 + R_0^2} - R_0$$

Where R_0 is a constant value behaving as a threshold.

Hyperbolic penalty function



Hyperbolic penalty function



Logarithm method

- Why do we need this?
  - We have to update minimum-phase and maximum-phase deconvolution filters respectively.
  - The new *u* variable can help us to separate minimum-phase and maximum-phase filters without crosstalk.

$$u = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0, u_1, u_2, u_3, \ldots), \quad z = e^{iw}$$

$$u^+ = (0, u_1, u_2, u_3, \ldots); \qquad u^- = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0)$$

$$U^+(z) = 0 + u_1 z + u_2 z^2 + \cdots; \qquad U^-(z) = 0 + u_{-1}/z + u_{-2}/z^2 + \cdots$$

$$F_a = e^{U^+}; \qquad F_b = e^{U^-}$$

$$u = u^+ + u^- \ \text{(No overlap between } u^+ \text{ and } u^- \text{)}$$

$$u = ([\text{maximum-phase part}], 0, [\text{minimum-phase part}])$$

Logarithm method

- Intuitive proof

$$u = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0, u_1, u_2, u_3, \ldots), \quad z = e^{iw}$$

$$u^+ = (0, u_1, u_2, u_3, \ldots);$$

$$U^+(z) = 0 + u_1 z + u_2 z^2 + \cdots;$$

$$F_a = e^{U^+};$$

$$u^- = (\ldots, u_{-3}, u_{-2}, u_{-1}, 0)$$

$$U^-(z) = 0 + u_{-1}/z + u_{-2}/z^2 + \cdots$$

$$F_b = e^{U^-}$$

$e^{U^+} = 1 + U^+ + (U^+)^2/2! + (U^+)^3/3!$ The filter F_a is causal.

$e^{-(U^+)} = 1 - U^+ + (U^+)^2/2! - (U^+)^3/3!$ The inverse F_a is also causal.

So the filter F_a is minimum-phase.

Likewise, we can proof F_b is maximum-phase

Logarithm method

- How to implement the logarithm method?

We use a iterative gradient based inversion scheme. For each iteration, we need to know

I. The gradient (update direction for **u**), $\Delta \mathbf{u}$;
II. The update direction for residual **r**, $\Delta \mathbf{r}$;
III. The update step length α .

Logarithm method

- I. The gradient (update direction for **u** in each iteration)

$$J = \mathrm{Hyp}(\mathbf{r}) = \sum_t H(r_t) = \sum_t H([\mathrm{IFT}(De^U)]_t)$$

$$\Delta\mathbf{u} = \frac{\partial J}{\partial \mathbf{u}} = \sum_t \frac{\partial H(r_t)}{\partial r_t}\frac{\partial r_t}{\partial \mathbf{u}} = \sum_t H'(r_t)\frac{\partial r_t}{\partial \mathbf{u}}$$

If we look at the τ-th component of $\Delta\mathbf{u}$

$$\Delta u_\tau = [\Delta\mathbf{u}]_\tau = \sum_t H'(r_t)\frac{\partial r_t}{\partial u_\tau}$$

$$\frac{\partial r_t}{\partial u_\tau} = \frac{\partial[\mathrm{IFT}(De^U)]_t}{\partial u_\tau} = [\mathrm{IFT}(De^U\frac{\partial U}{\partial u_\tau})]_t$$

Logarithm method

- I. The gradient (update direction for **u** in each iteration)

$$\frac{\partial r_t}{\partial u_\tau} = \frac{\partial[\mathrm{IFT}(De^U)]_t}{\partial u_\tau} = [\mathrm{IFT}(De^U \frac{\partial U}{\partial u_\tau})]_t$$

$$\because U = \cdots + u_{-2}/z^2 + u_{-1}/z + u_0 + u_1 z + u_2 z^2 + \cdots$$

$$\frac{\partial U}{\partial u_\tau} = z^\tau$$

$$\therefore \frac{\partial r_t}{\partial u_\tau} = [\mathrm{IFT}(De^U z^\tau)]_t = r_{t+\tau}$$

Logarithm method

- 1. The gradient (update direction for **u** in each iteration)

$$\Delta u_\tau = [\Delta \mathbf{u}]_\tau = \sum_t H'(r_t)\frac{\partial r_t}{\partial u_\tau} = \sum_t H'(r_t) r_{t+\tau}$$

$$\Delta \mathbf{u} = \mathbf{r} \odot H'(\mathbf{r})$$

($\odot$ denotes cross-correlation)

Logarithm method

- II. The update direction for residual **r**

$$\mathbf{r} + \alpha\Delta\mathbf{r} = \mathrm{IFT}(De^{U+\alpha\Delta U}) = \mathrm{IFT}(De^{U}e^{\alpha\Delta U})$$

$$= \mathrm{IFT}(De^{U})\mathrm{IFT}(e^{\alpha\Delta U})$$

$$\mathrm{IFT}(e^{\alpha\Delta U}) = \mathrm{IFT}(e^{\alpha(\cdots+\Delta u_{-1}/z+0+\Delta u_1 z+\cdots)})$$

$$= \mathrm{IFT}(1+\alpha(\cdots+\Delta u_{-1}/z+0+\Delta u_1 z+\cdots)+\alpha^2(\cdots)+\cdots)$$

If we ignore higher terms of α (assuming α is small)

$$\mathrm{IFT}(e^{\alpha\Delta U}) = \mathrm{IFT}(1+\alpha(\cdots+\Delta u_{-1}/z+0+\Delta u_1 z+\cdots))$$

$$= \cdots,\alpha\Delta u_{-1},1,\alpha\Delta u_1,\cdots$$

Logarithm method

- II. The update direction for residual **r**

$$\mathbf{r} + \alpha\Delta\mathbf{r} = \mathrm{IFT}(De^{U+\alpha\Delta U}) = \mathrm{IFT}(De^{U} e^{\alpha\Delta U})$$

$$\mathbf{r} + \alpha\Delta\mathbf{r} = \mathbf{r} * (\cdots, \alpha\Delta u_{-1}, 1, \alpha\Delta u_{1}, \cdots)$$

$$= \mathbf{r} + \alpha\mathbf{r} * \Delta\mathbf{u}$$

$$\Delta\mathbf{r} = \mathbf{r} * \Delta\mathbf{u}$$

Logarithm method

- III. The update step length α
  - To find the update step length α , we try to minimize the object function by tuning α

$$\underset{\alpha}{\operatorname{argmin}}[\operatorname{Hyp}(\mathbf{r}+\alpha\Delta\mathbf{r})]$$

$$\frac{\partial \operatorname{Hyp}(\mathbf{r}+\alpha\Delta\mathbf{r})}{\partial\alpha}=0$$

  - We use Newton iteration to find this minimum.

$$\operatorname{Hyp}(\mathbf{r}+\alpha\Delta\mathbf{r})=\sum_t\left(H(r_t)+\alpha\Delta r_t H'(r_t)+(\alpha\Delta r_t)^2 H''(r_t)/2!+\cdots\right)$$

Ignore the terms higher than 2^{nd} order

Logarithm method

- III. The update step length α

$$\mathrm{Hyp}(\mathbf{r}+\alpha\Delta\mathbf{r}) = \sum_t \left(H(r_t) + \alpha\Delta r_t H'(r_t) + (\alpha\Delta r_t)^2 H''(r_t)/2! \right)$$

$$\frac{\partial\, \mathrm{Hyp}(\mathbf{r}+\alpha\Delta\mathbf{r})}{\partial\alpha} = \frac{\partial}{\partial\alpha} \sum_t \left(H(r_t) + \alpha\Delta r_t H'(r_t) + (\alpha\Delta r_t)^2 H''(r_t)/2! \right)$$

$$\alpha = -\frac{\sum_t \Delta r_t H'(r_t)}{\sum_t (\Delta r_t)^2 H''(r_t)}$$

Because we use Newton method here (ignoring higher order terms in Taylor expansion), we need a iteration to get final α.

Logarithm method

- The iteration to get final alpha

$\alpha = 0$

loop

{

$$\alpha_{inc} = -\frac{\sum_t \Delta r_t H'(r_t)}{\sum_t (\Delta r_t)^2 H''(r_t)}$$

$$\alpha = \alpha + \alpha_{inc}$$

$$r_t = r_t + \alpha_{inc} \Delta r_t$$

}

Logarithm method

- We can use trial and error method to reduce the over shoot problem

loop{

$$\alpha_{inc} = -\frac{\sum_t \Delta r_t H'(r_t)}{\sum_t (\Delta r_t)^2 H''(r_t)}$$

loop {

$$\mathbf{r}_{temp} = \mathbf{r} + \alpha_{inc} \Delta \mathbf{r}$$

If $(\text{Hyp}(\mathbf{r}_{temp}) > \text{Hyp}(\mathbf{r}))$ then $(\alpha_{inc} = \alpha_{inc} / 2)$ Else Break

}

$$\alpha = \alpha + \alpha_{inc}$$

$$\mathbf{r} = \mathbf{r} + \alpha_{inc} \Delta \mathbf{r}$$

}

The magic of Logarithm method

- The magic of exponential operator "e"
  - Why gradient $\Delta \boldsymbol{u}$ is a function of shifted output $\boldsymbol{r}$ is improtant?

$$\Delta u_\tau = [\Delta \mathbf{u}]_\tau = \sum_t H'(r_t) \frac{\partial r_t}{\partial u_\tau} = \sum_t H'(r_t) r_{t+\tau}$$

  - The gradient should be vanished in the final solution. If $H(r_t)$ is convention L^2 norm rather than hyperbolic penalty function,

$$H'(r_t) = (r_t^2)' = 2r_t$$

$$0 = \Delta u_\tau = \sum_t H'(r_t) r_{t+\tau} = \sum_t 2 r_t r_{t+\tau}$$

  - The auto-correlation of the output is 0 except at the origin. The shifted output is orthogonal with output itself. The says the output is white.

The magic of Logarithm method

- The magic of exponential operator "e"
  - If we do not have "e" here, the gradient is a function of shifted input ***d***

$$\Delta f_\tau = [\Delta \mathbf{f}]_\tau = \sum_t H'(r_t)\frac{\partial r_t}{\partial f_\tau} = \sum_t H'(r_t) d_{t+\tau}$$

  - The gradient should be vanished in the final solution. If $H(r_t)$ is convention L^2 norm rather than hyperbolic penalty function,

$$H'(r_t) = (r_t^2)' = 2r_t$$

$$0 = \Delta f_\tau = \sum_t H'(r_t) d_{t+\tau} = \sum_t 2 r_t d_{t+\tau}$$

  - The output is orthogonal with shifted input. The says the output is not white anymore.