Lecture 4

Cadzow/SSA filters
(Singular Spectrum Analysis)

M D Sacchi
University of Alberta

Course information

- Email: msacchi@ualberta.ca

- https://github.com/msacchi/SEP_lectures

First, Rank-Reduction

- Rank: Number of independent columns (or rows) of a matrix
- SVD: Is a rank-revealing algorithm, number of singular values different from zero is the rank
- Effective rank: Number of singular values above a given threshold
- SVD: Singular Value Decomposition, an algorithm for matrix decomposition, matrix compression, computing the pseudo-inverse, etc…

Eckard-Young (1936) theorem

The problem:

Given $\mathbf{A}$ of size $M \times N$
Find the rank-p matrix $\mathbf{A}_p$
that minimizes $J = \|\mathbf{A} - \mathbf{A}_p\|_F$

The solution:

$$\mathbf{U}, \mathbf{S}, \mathbf{V} = \text{svd}[\mathbf{A}] \rightarrow$$
$$\mathbf{A}_p = \mathbf{U}_p \mathbf{S}_p \mathbf{V}_p^H$$
$$= \mathbf{U}_p \mathbf{U}_p^H \mathbf{A}$$

C. Eckart, G. Young, The approximation of one matrix by another of lower rank. Psychometrika, Volume 1, 1936

Eckard-Young theorem

- The SVD: $$\mathbf{U}, \mathbf{S}, \mathbf{V} = \text{svd}[\mathbf{A}] \rightarrow$$ $$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^H$$

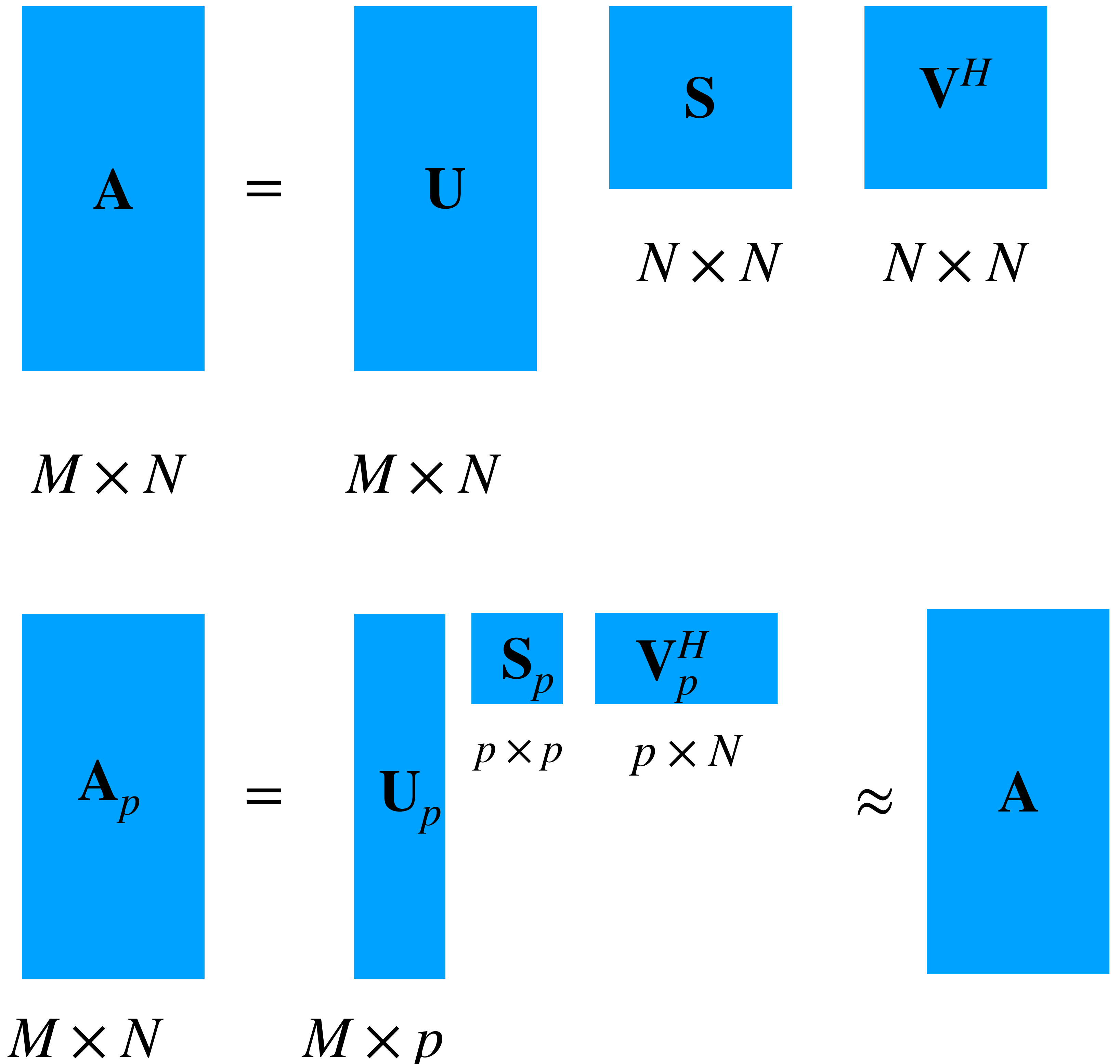
- where $$\mathbf{U}^H\mathbf{U} = \mathbf{I}_N, \quad \mathbf{V}^H\mathbf{V} = \mathbf{I}_N$$

- The columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal vectors: $$\mathbf{u}_k^H\mathbf{u}_k = \delta_{i,j}\,, \quad \mathbf{v}_k^H\mathbf{v}_k = \delta_{i,j}$$

- $\mathbf{S}$ is the diagonal matrix of singular values $$\mathbf{S} = \text{diag}(s_1, s_2, \ldots, s_N) \text{ with } s_1 \geq s_2 \geq s_3 \ldots \geq s_N$$

The decomposition

The approximation



Compression

$$\mathbf{A} : M \times N$$

$$\mathbf{A}_p : M \times N$$

$$\mathbf{U}_p : M \times p$$

$$\mathbf{V}_p : N \times p$$

$\mathbf{S}_p : p \times p$ is diagonal only p values are saved

$$C = \frac{M \times p + N \times p + p}{M \times N} << 1 \text{ if } p \text{ small}$$

The approximation:

$$\mathbf{U}, \mathbf{S}, \mathbf{V} = \textsf{svd}[\mathbf{A}] \rightarrow$$
$$\mathbf{A}_p = \mathbf{U}_p \mathbf{S}_p \mathbf{V}_p^H$$
$$= \mathbf{U}_p \mathbf{U}_p^H \mathbf{A}$$

Can be written as the sum of rank-1 matrices:

$$\mathbf{A}_p = \sum_{k=1}^{p} s_k \mathbf{u}_k \mathbf{v}_k^H = \sum_{k=1}^{p} s_k \mathbf{E}_k$$

Eigen-images are rank-1 matrices given by: $\mathbf{E_k} = \mathbf{u}_k \mathbf{v}_k^H$

Approximation error: $\epsilon_p = \|\mathbf{A} - \mathbf{A}_p\|_F = \sum_{k>p} \sigma_k \mathbf{E}_k$

Sergio L. M. Freire, Tad J. Ulrych; Application of singular value decomposition to vertical seismic profiling. *Geophysics* 1988;; 53 (6): 778–785

Example

1) $D(t,x)$ **is a matrix consisting of events with no dip**
2) $D(t,x)$ **is a matrix consisting of parabolic events (variable dip)**

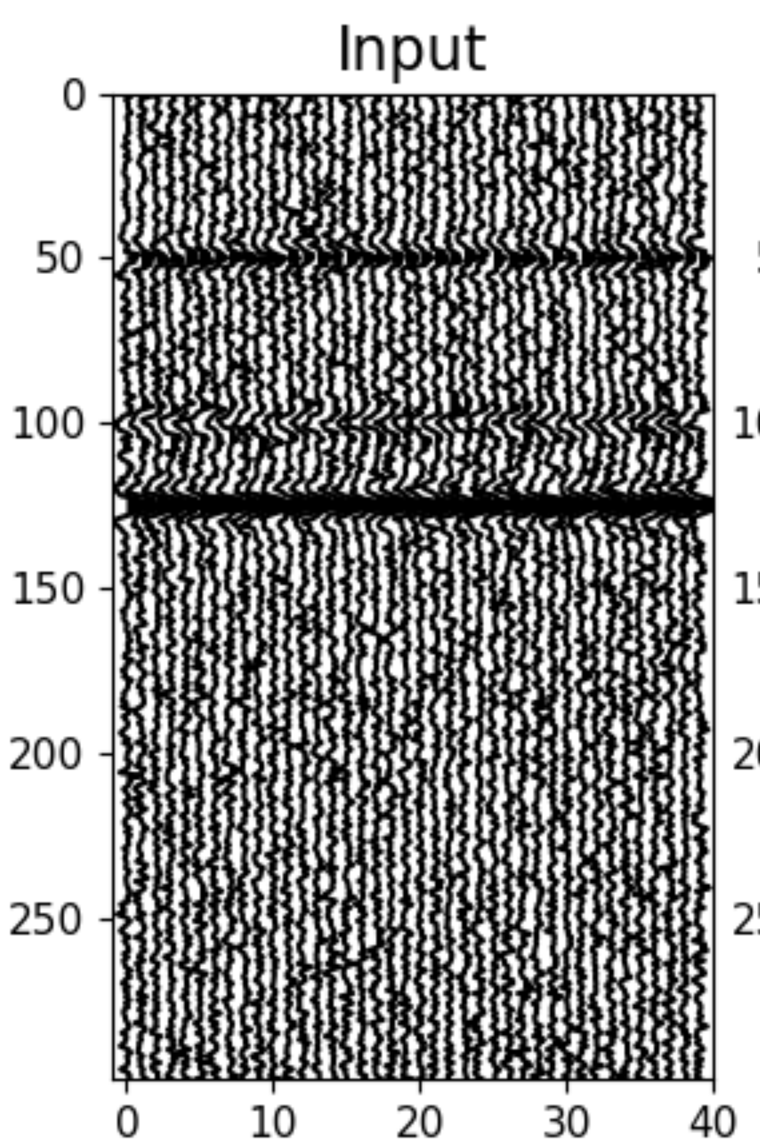
We will see the impact of Eigen-image filtering in 1) and 2)

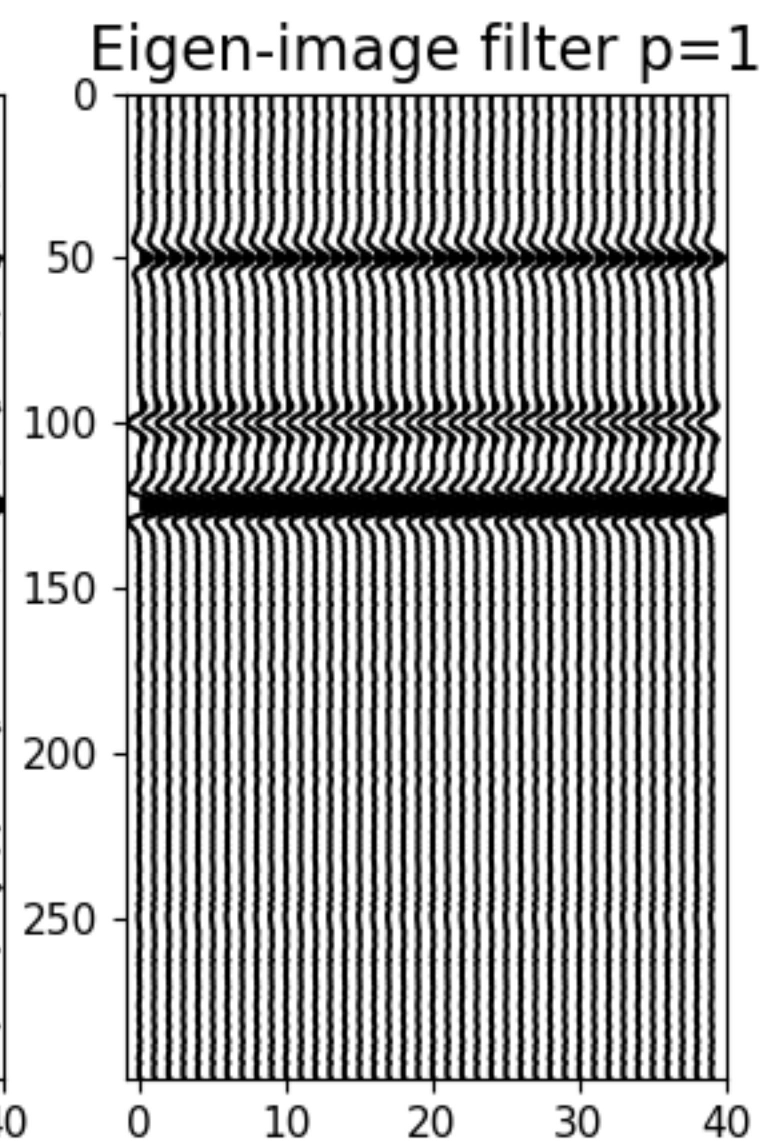
```
F = svd(D)
p = 1
U = (F.U)[:,1:p]
Df = U*U'*D
```

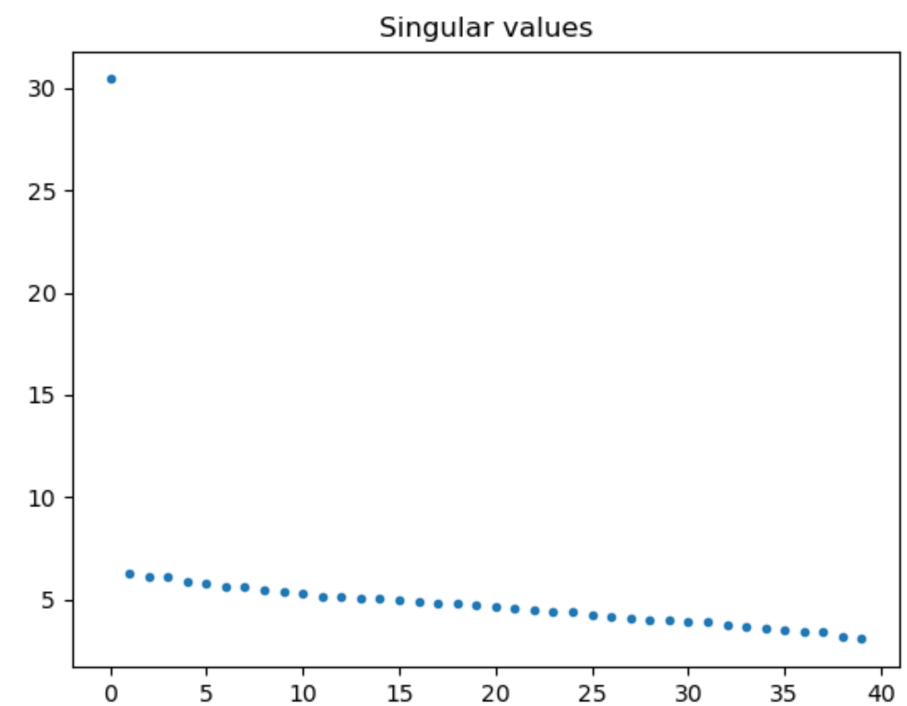
`Test_svd_eigen.ipynb`

Simple idea suppress noise by rank-reduction

- Assume ideal data can be represented by a matrix of rank p
- Use SVD approximation as a filter directly applied to data in t-x. This is too naive, it assumes horizontal events



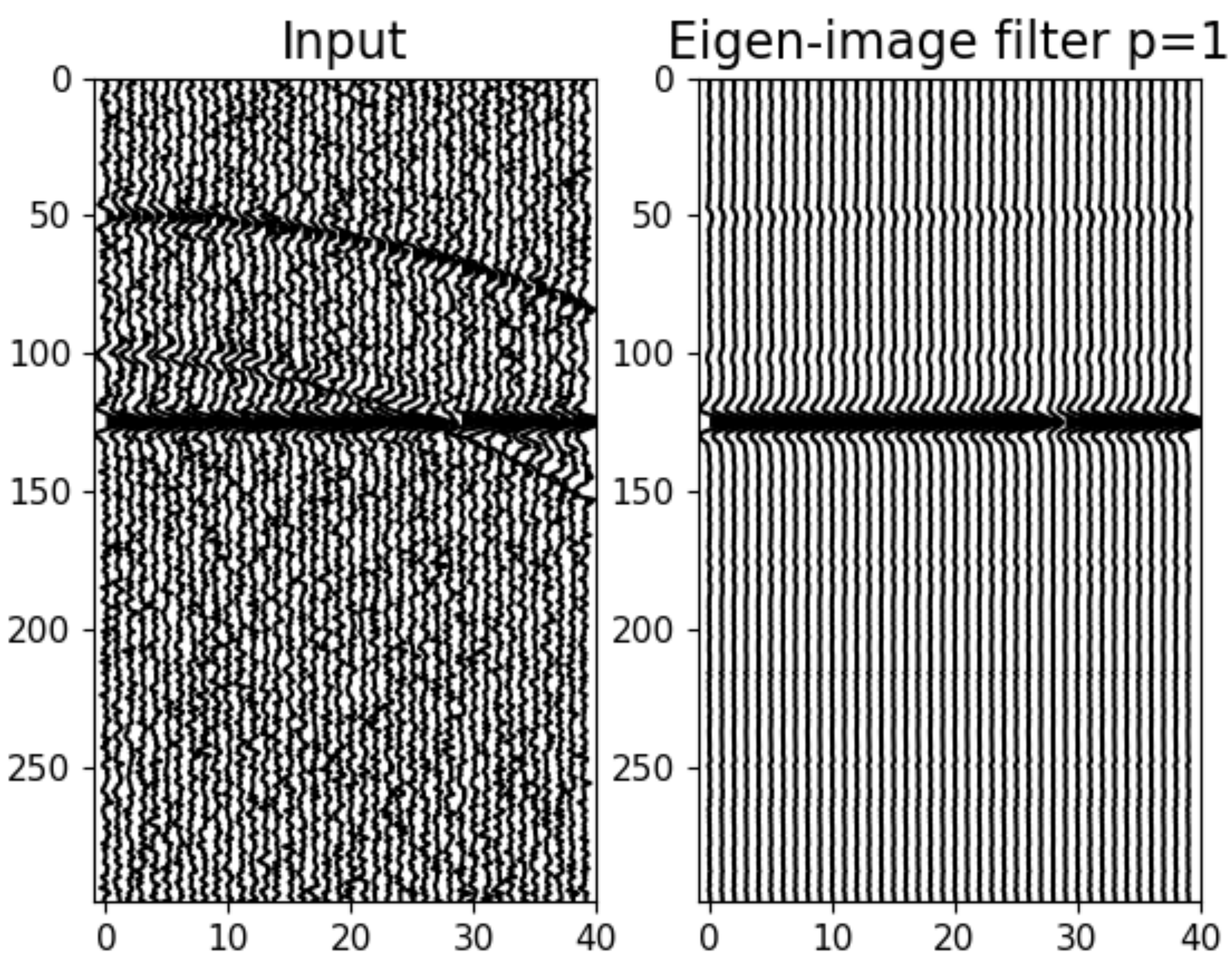


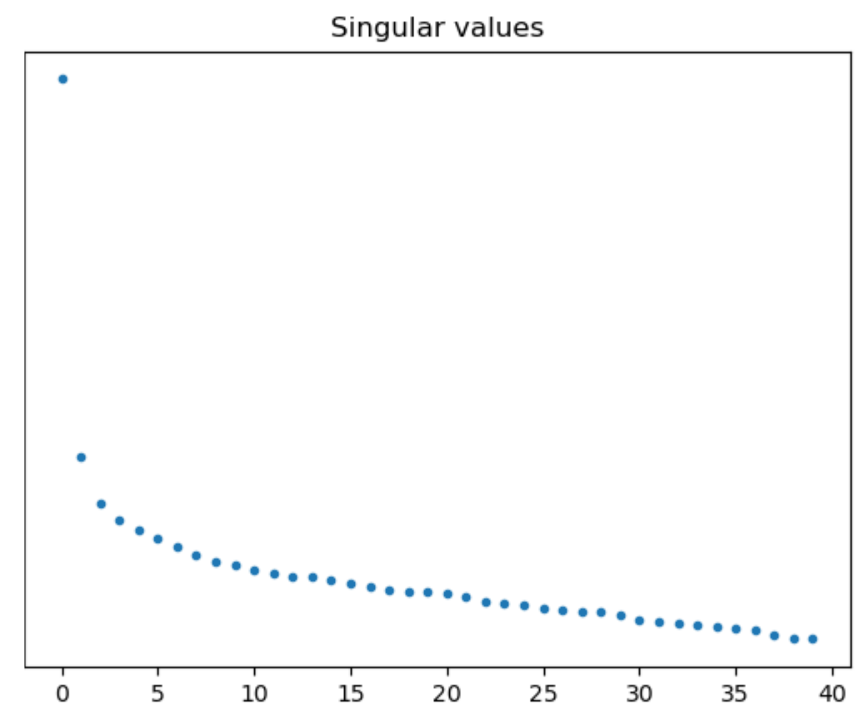


Clear cut-off

Simple idea suppress noise by rank-reduction

- Assume ideal data can be represented by a matrix of rank p
- Use SVD approximation as a filter directly applied to data in t-x won’t work when there is dip (constant or as below varying dip)

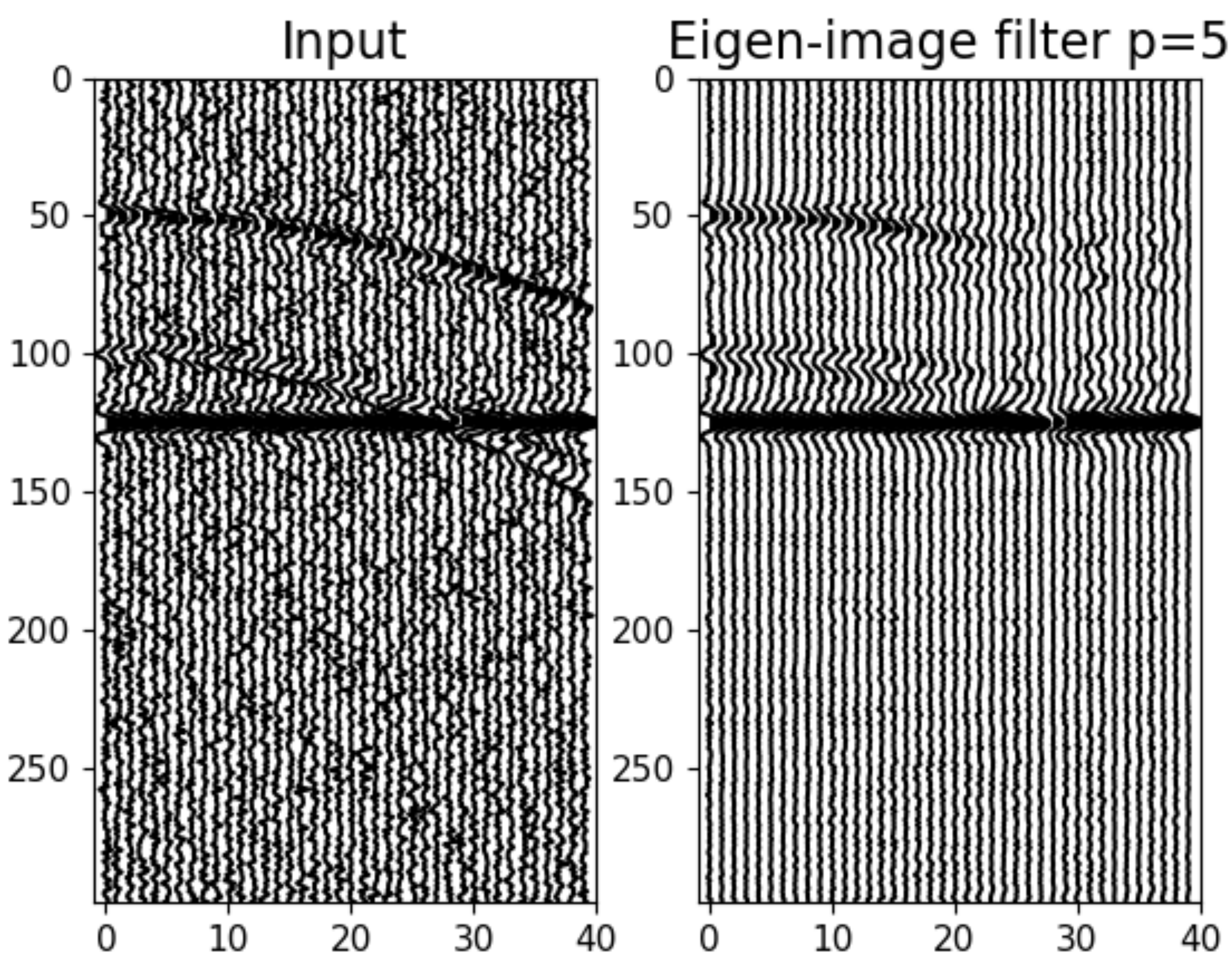


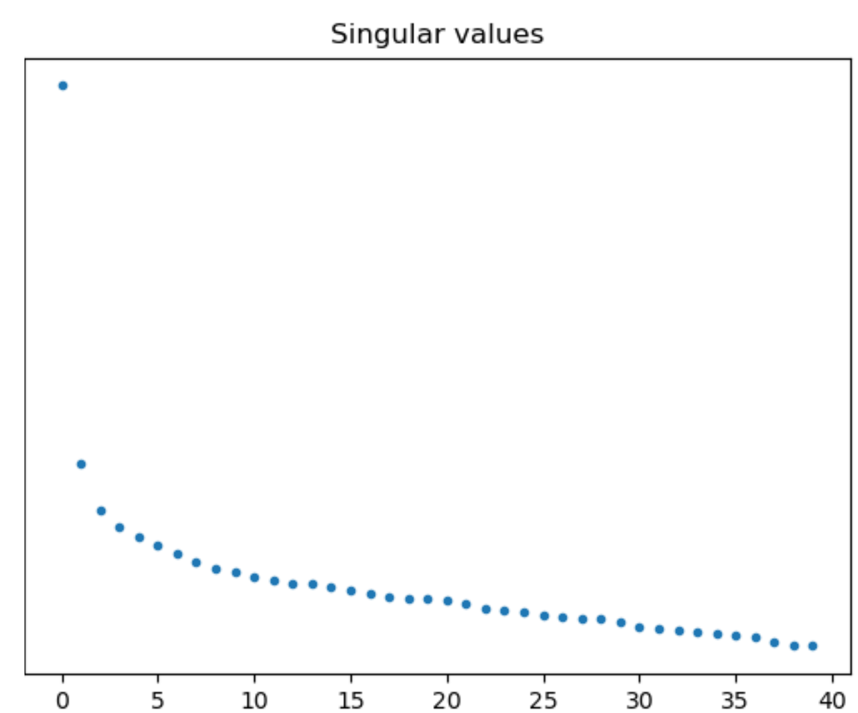


No clear cut-off

Simple idea suppress noise by rank-reduction

- Assume ideal data can be represented by a matrix of rank p
- Use SVD approximation as a filter directly applied to data in t-x won't work when there is dip (constant or as below varying dip)





Working in FX or FXY domain to cope with dips

- One linear dip in f-x

$$d(t,x) = a(t-px) \rightarrow D(\omega,x) = A(w)e^{i\omega px}$$

- One 2D plane wave in f-xy

$$d(t,x,y) = a(t-px-qy) \rightarrow D(\omega,x) = A(\omega)e^{i\omega px}e^{i\omega qx}$$

$$D(\omega,x,y) = A(w)e^{i\omega px}e^{i\omega qx} \rightarrow \mathbf{D}(\omega) = \mathbf{a}(\omega)\mathbf{b}(\omega)^H$$

rank=1

Stewart R. Trickett; *F-xy* eigenimage noise suppression. *Geophysics* 2003;; 68 (2): 751–759.

doi: https://doi.org/10.1190/1.1567245

SSA filters (Singular Spectrum Analysis)

- F-xy eigen-images (Trickett, 2003) is restricted to 2D cubes and is basically applying SVD eigen-decomposition to constant frequency slices of data in x and y

- A more interested and generalizable approach entails embedding f-x signals (or f-x-y signals) into Hankel matrices

- The latter leads to a series of methods for denoising and reconstruction called SSA or Cadzow filtering/reconstruction

- This idea was rediscovered many times: SSA, Cadzow, Caterpillar method, Matrix Pencil method

SSA development

- Consider a harmonic signal of wavenumber k:

$$S_n = Ae^{ikn}, n = 0 \ldots N-1$$

- Then

$$S_{n-1} = Ae^{ik(n-1)} = Ae^{ikn}\, e^{-ia}$$

- Which leads to

$$\boxed{S_n = aS_{n-1}}$$

SSA development

Consider a harmonic signal of length 5:

$$S = [S_1, S_2, S_3, S_4, S_5]^T$$

We embed the signal into Hankel matrix and use $S_n = aS_{n-1}$

$$N_r = [L/2 + 1] = 3\,, \qquad N_c = L - N_r + 1 = 3$$

$$\mathbf{H}(\mathbf{s}) = \begin{pmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{pmatrix} = \begin{pmatrix} S_1 & aS_1 & a^2S_1 \\ S_2 & aS_2 & a^2S_2 \\ S_3 & aS_3 & a^2S_3 \end{pmatrix}$$

The Hankel matrix is rank 1

SSA development

- Consider a sum of p harmonic signals:

$$S_n = \sum_{j=1}^{P} A_j e^{ik_j n}, n = 0 \ldots N-1$$

- Then $S_n = a_1 S_{n-1} + a_2 S_{n-2} + \ldots a_p S_{n-p}$ and

$$\mathbf{H}(\mathbf{s}) = \begin{pmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{pmatrix}$$ **is a rank p matrix**

SSA development

- Antidiagonal averaging operator

$$\mathscr{A}[\mathbf{H}(\mathbf{s})] = \mathscr{A}\begin{pmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{pmatrix}$$

SSA development

- The Algorithm

$\mathbf{s}$: input 1D signal

$\hat{\mathbf{s}}$: output 1D signal

$$\hat{\mathbf{s}} = \mathscr{A}\,\mathscr{R}\,\mathbf{H}(\mathbf{s})$$

Create Hankel matrix

Rank reduction

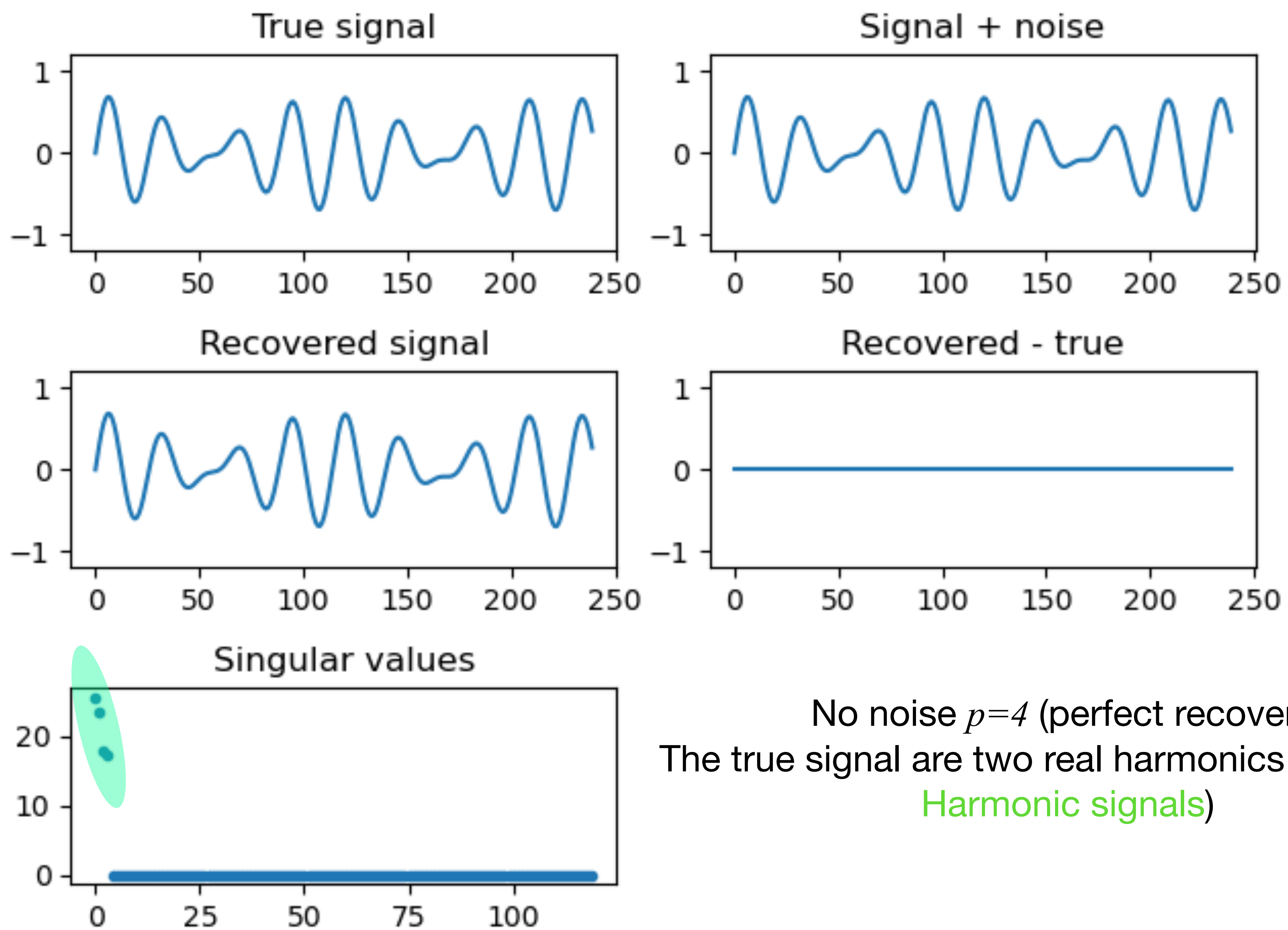
Antidiagonal averaging

SSA development

```
# SSA filter
   p = 4
   H = create_hankel_matrix(s_in)
   F = svd(H)
   U = (F.U)[:,1:p]
   Hp = U*U'*H
   s_out = average_antidiagonals(Hp)
```

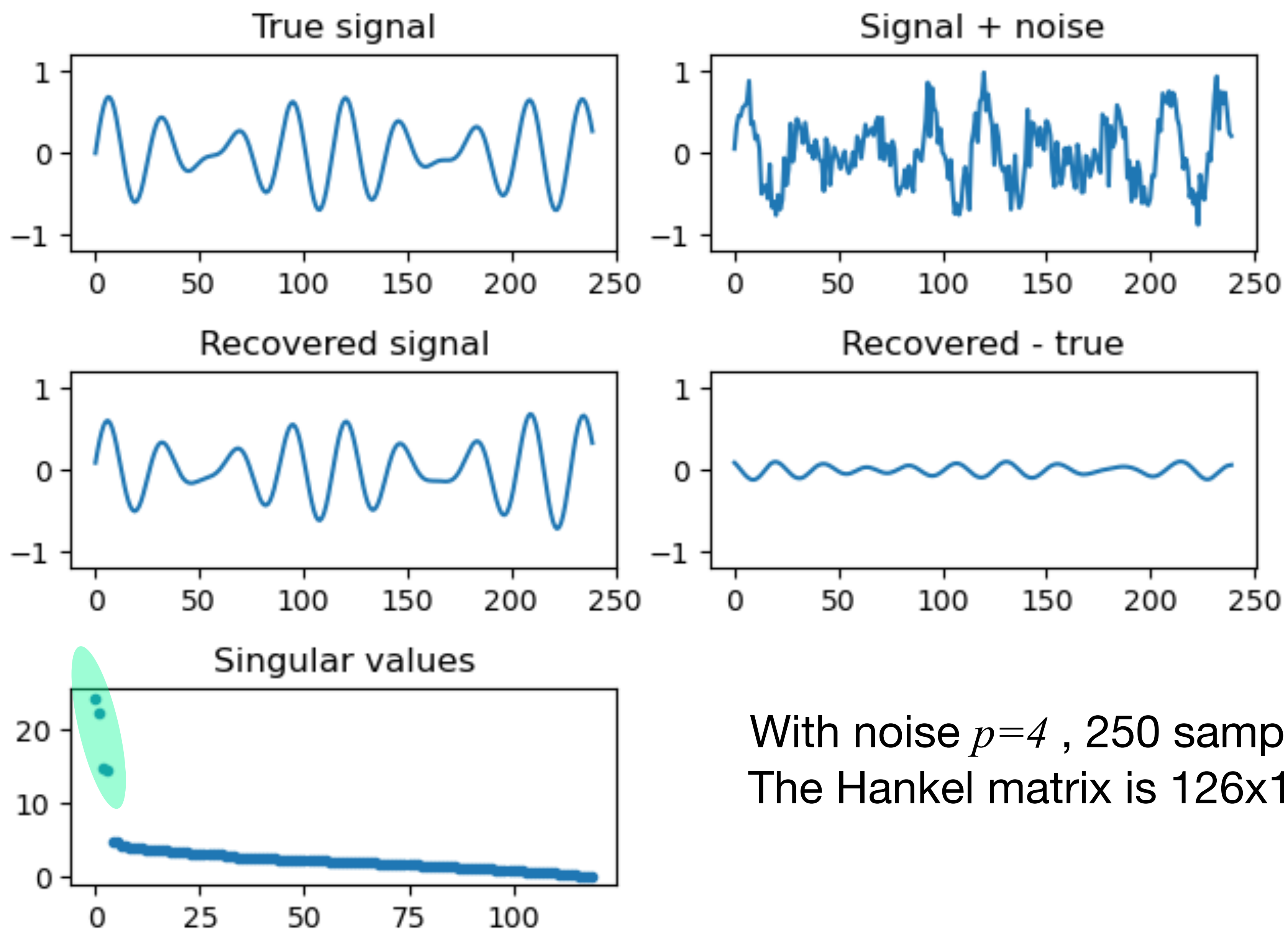
`Test_1D_SSA.ipynb`

SSA development



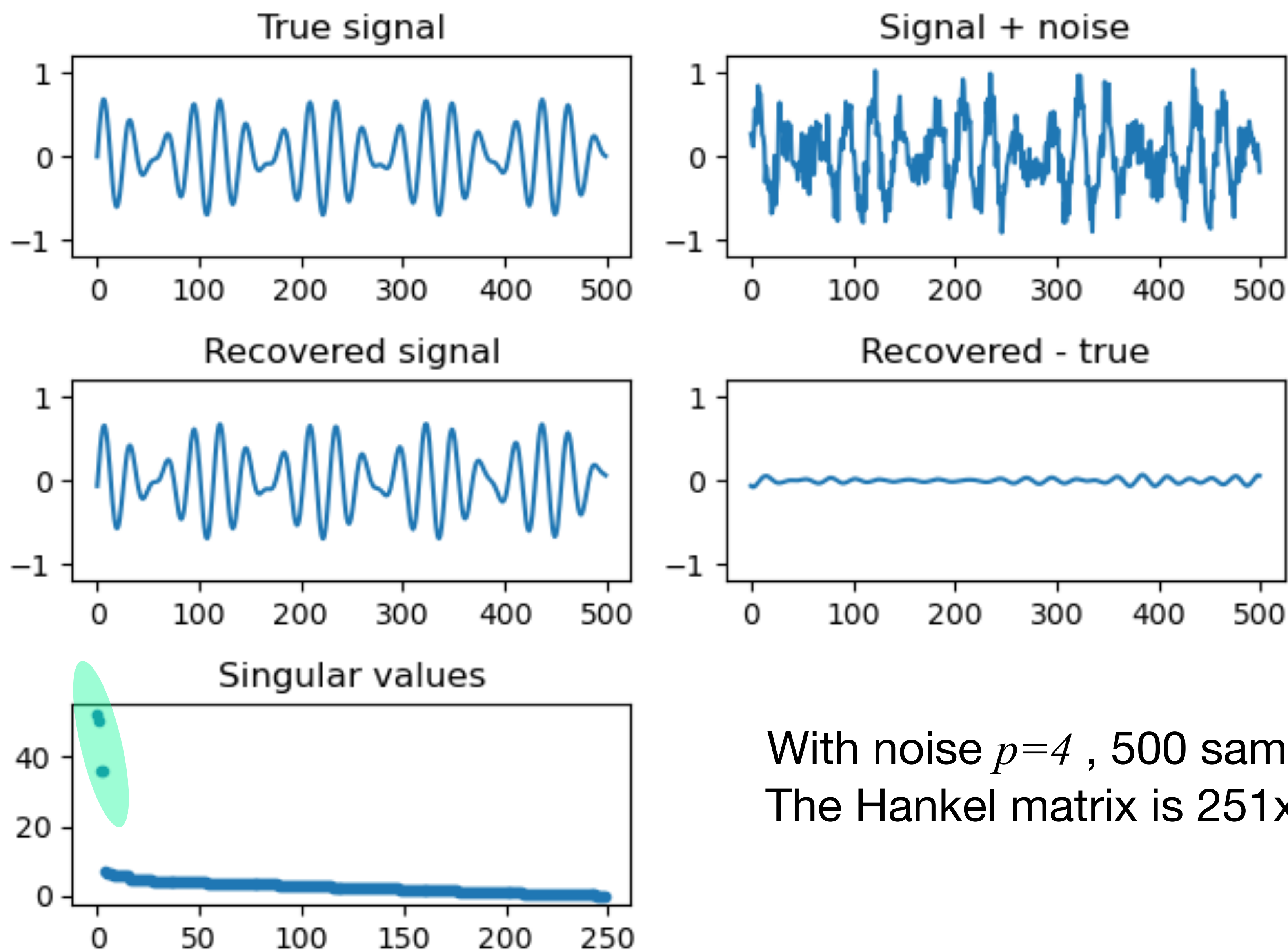
No noise $p=4$ (perfect recovery)
The true signal are two real harmonics (4 complex Harmonic signals)

SSA development



With noise $p=4$, 250 samples
The Hankel matrix is 126x125

SSA development



With noise p=4 , 500 samples

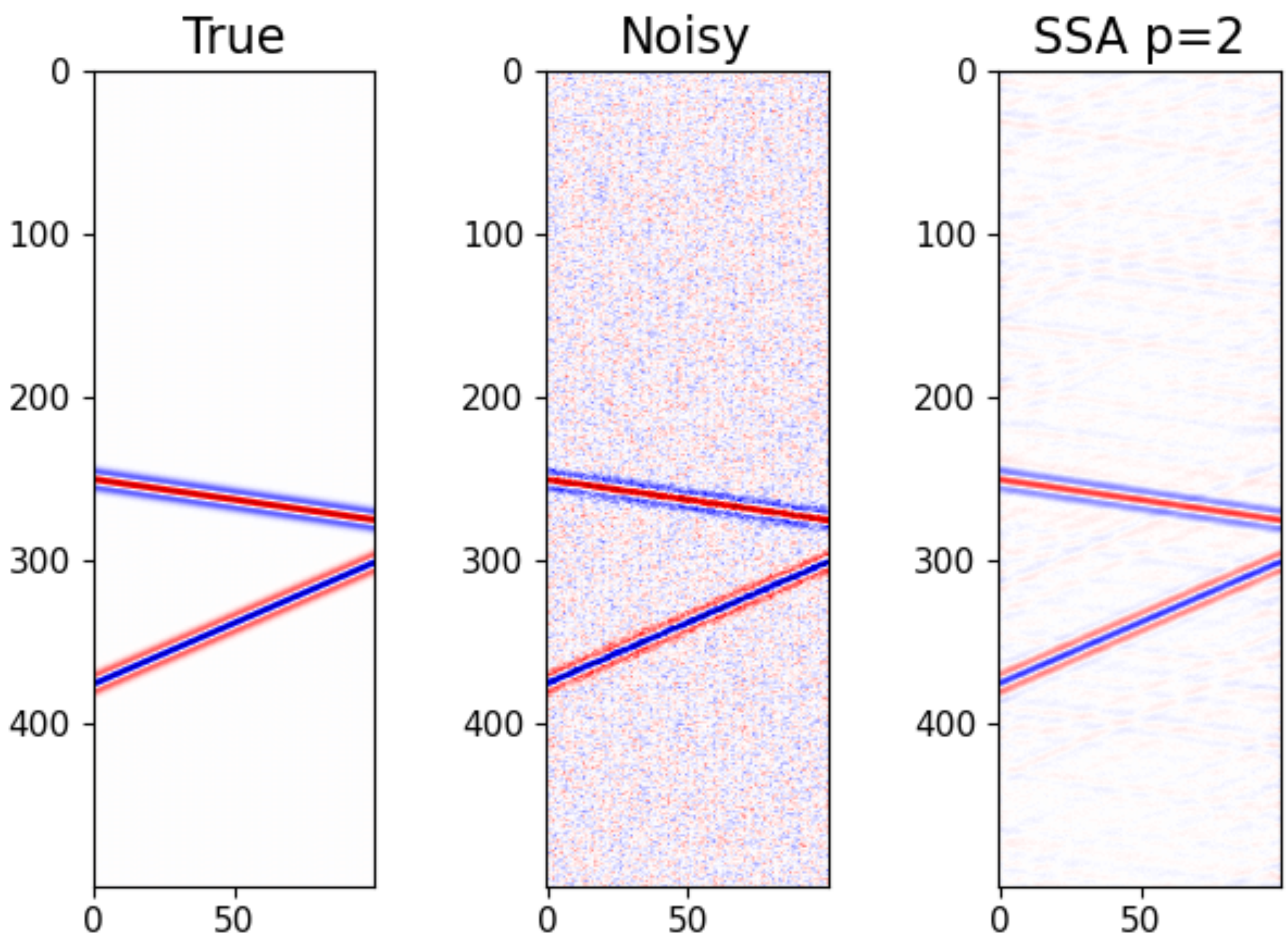
The Hankel matrix is 251x250

2D - SSA in FX domain

- The 1D SSA algorithm is applied to signals in the f-x domain.
- In this case, we apply SSA to constant frequency signals that vary in space (similar to classical FX deconvolution)
- This is a 2D (2D SSA) application but the SSA is 1D in space (this might lead to confusion when dealing with N-D signals)
    - For instance, 5D SSA reconstruction means 4D SSA spatial reconstruction on frequency slices

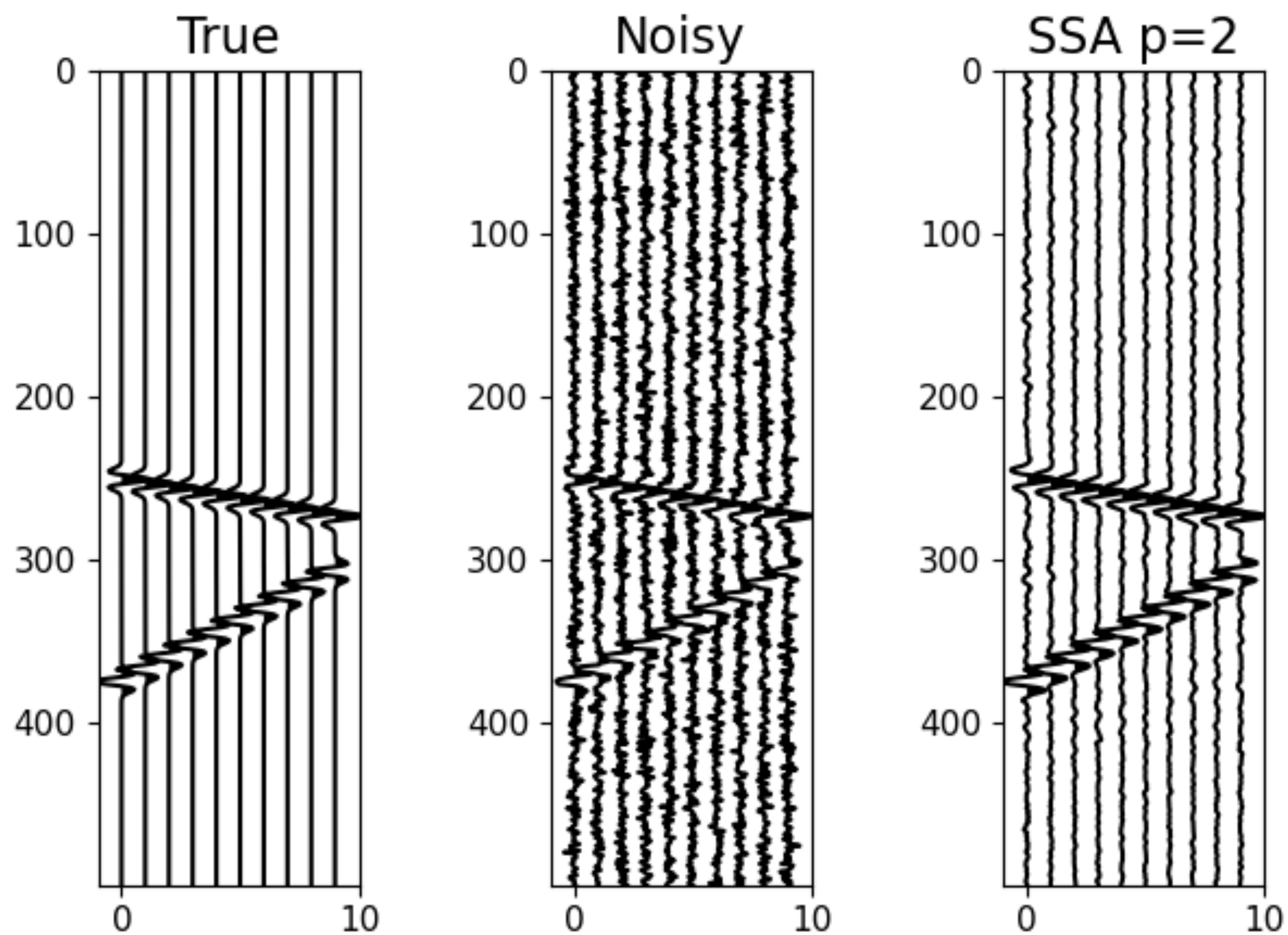
2D - SSA in FX domain

`Test_SSA.ipynb`



2D - SSA in FX domain

`Test_SSA.ipynb`



Every 10 traces

3D SSA for data in f-xy

- 3D t-xy cube: $s(t,x,y) \rightarrow S(\omega,x,y) \rightarrow \mathbf{S}(\omega) \in C^{N_x \times N_y}$
- Elements are given by $S_{l,j}\,, \quad l = 1 \ldots N_x\,, \quad j = 1 \ldots N_y$
- Assume a 5 x 5 spatial grid

$$\mathbf{H}_j = \begin{pmatrix} S_{1,j} & S_{2,j} & S_{3,j} \\ S_{2,j} & S_{3,j} & S_{4,j} \\ S_{3,j} & S_{4,j} & S_{5,j} \end{pmatrix}, j = 1:5 \rightarrow \qquad \mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 \\ \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_4 \\ \mathbf{H}_3 & \mathbf{H}_4 & \mathbf{H}_5 \end{pmatrix}$$

Form 5 Hankel Matrices

Embed them into a level-2 Block Hankel Matrix

3D SSA for data in f-xy

$\mathbf{S}$: input 2D signal for constant f slide

$\hat{\mathbf{S}}$: output 2D signal for constant f slide

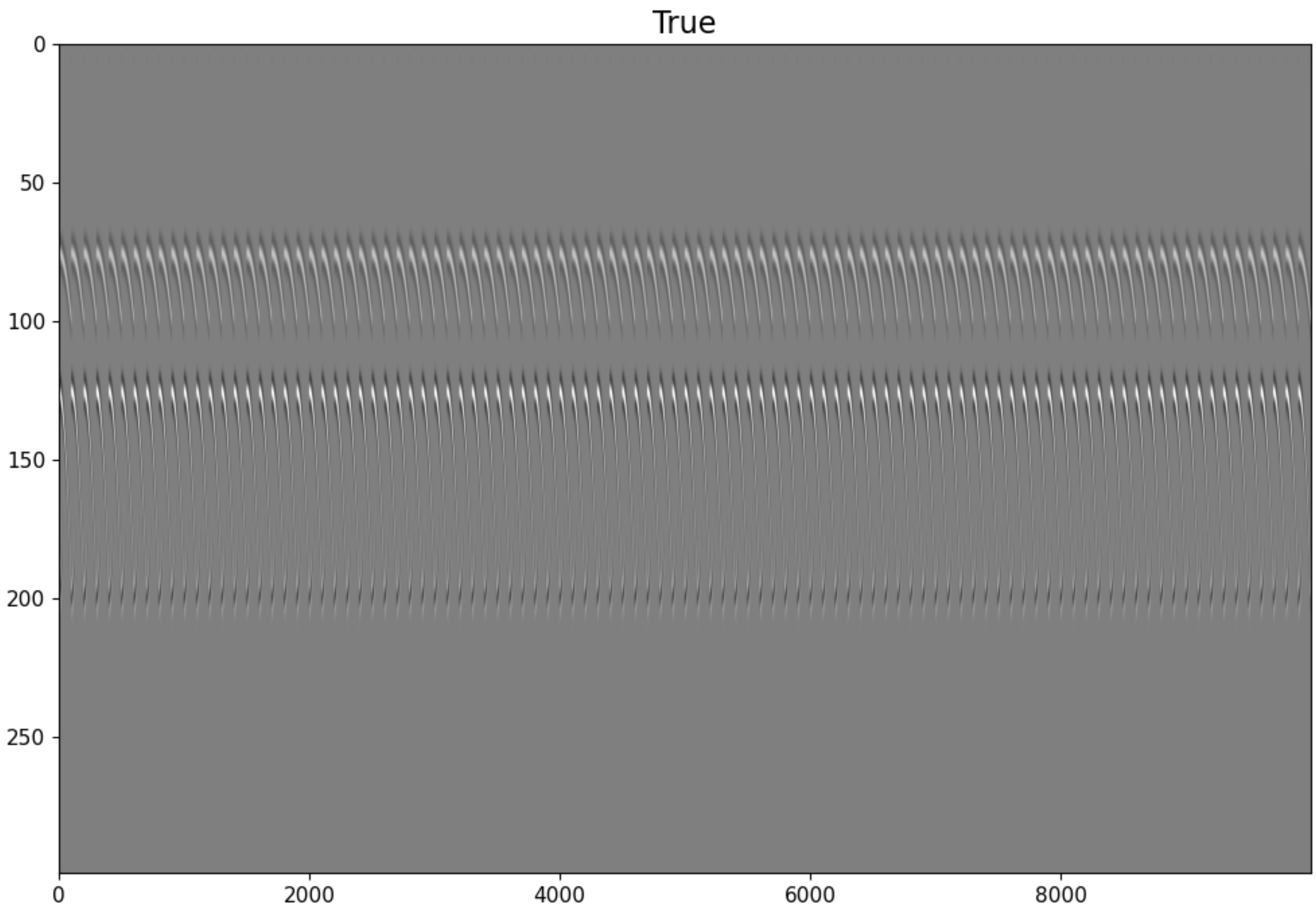
$$\hat{\mathbf{S}} = \mathscr{A}\,\mathscr{R}\,\mathbf{H}(\mathbf{S})$$

Create a level-2 Block Hankel matrix

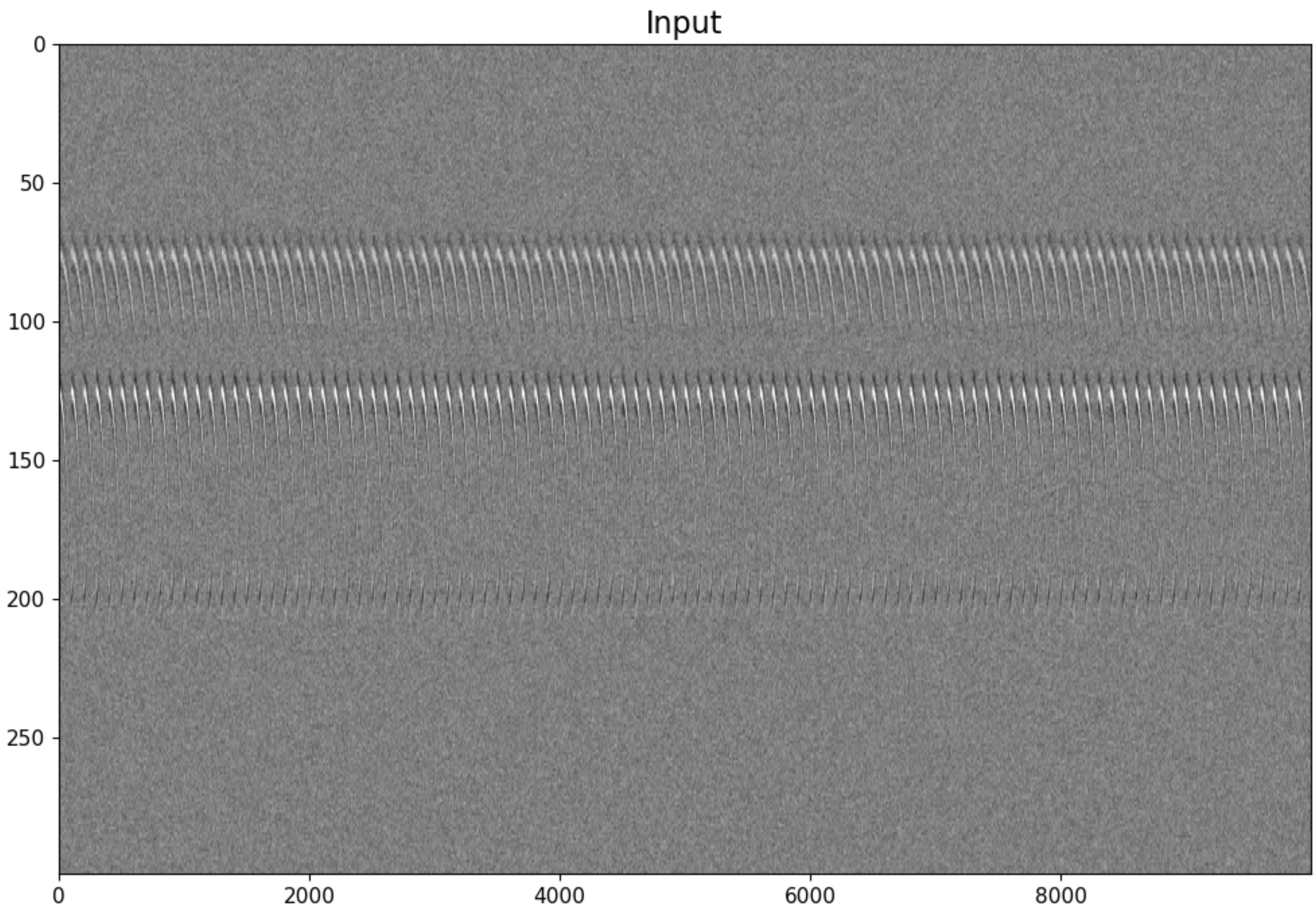
Rank reduction

Antidiagonal block averaging

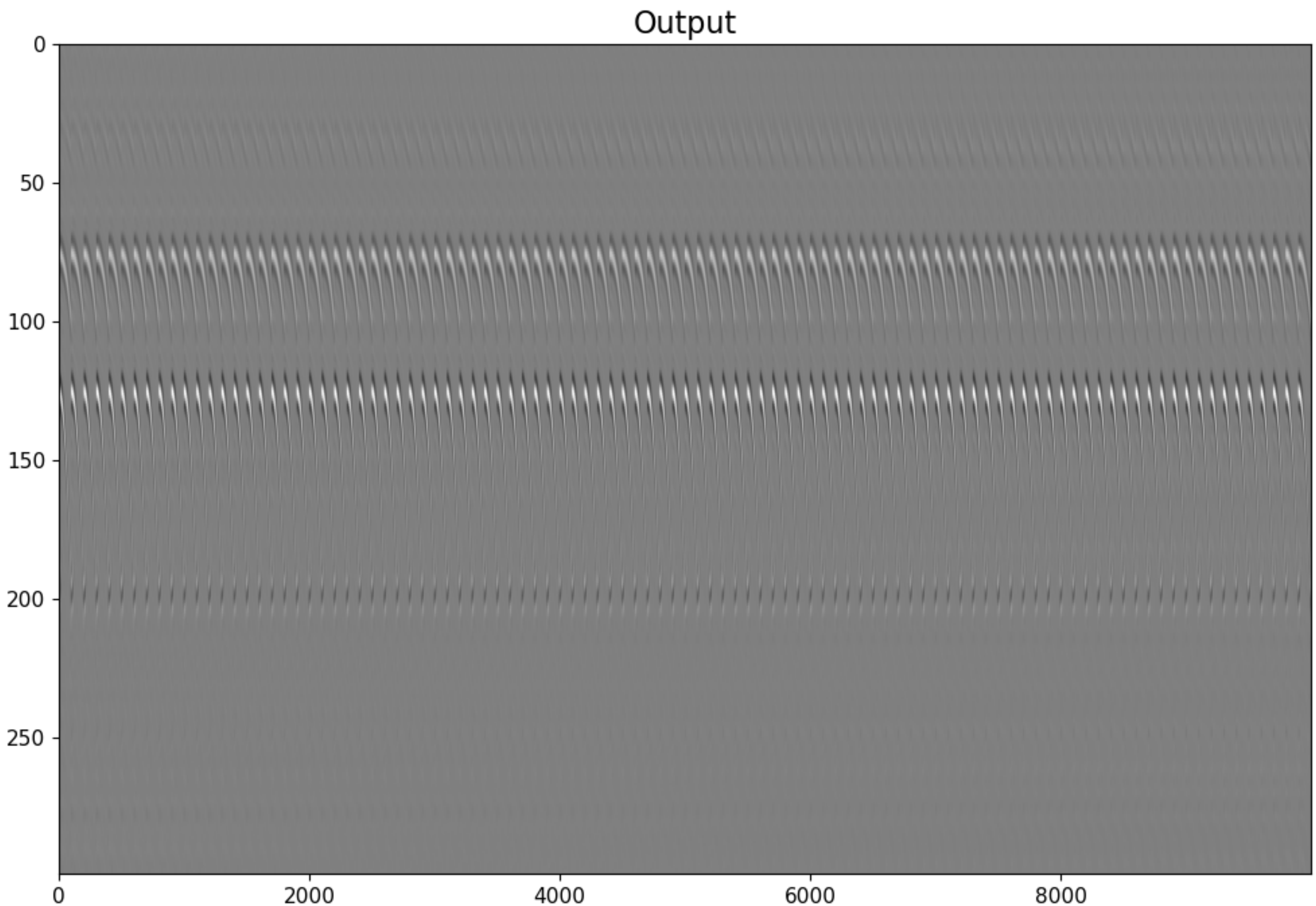
3D SSA for data in f-xy



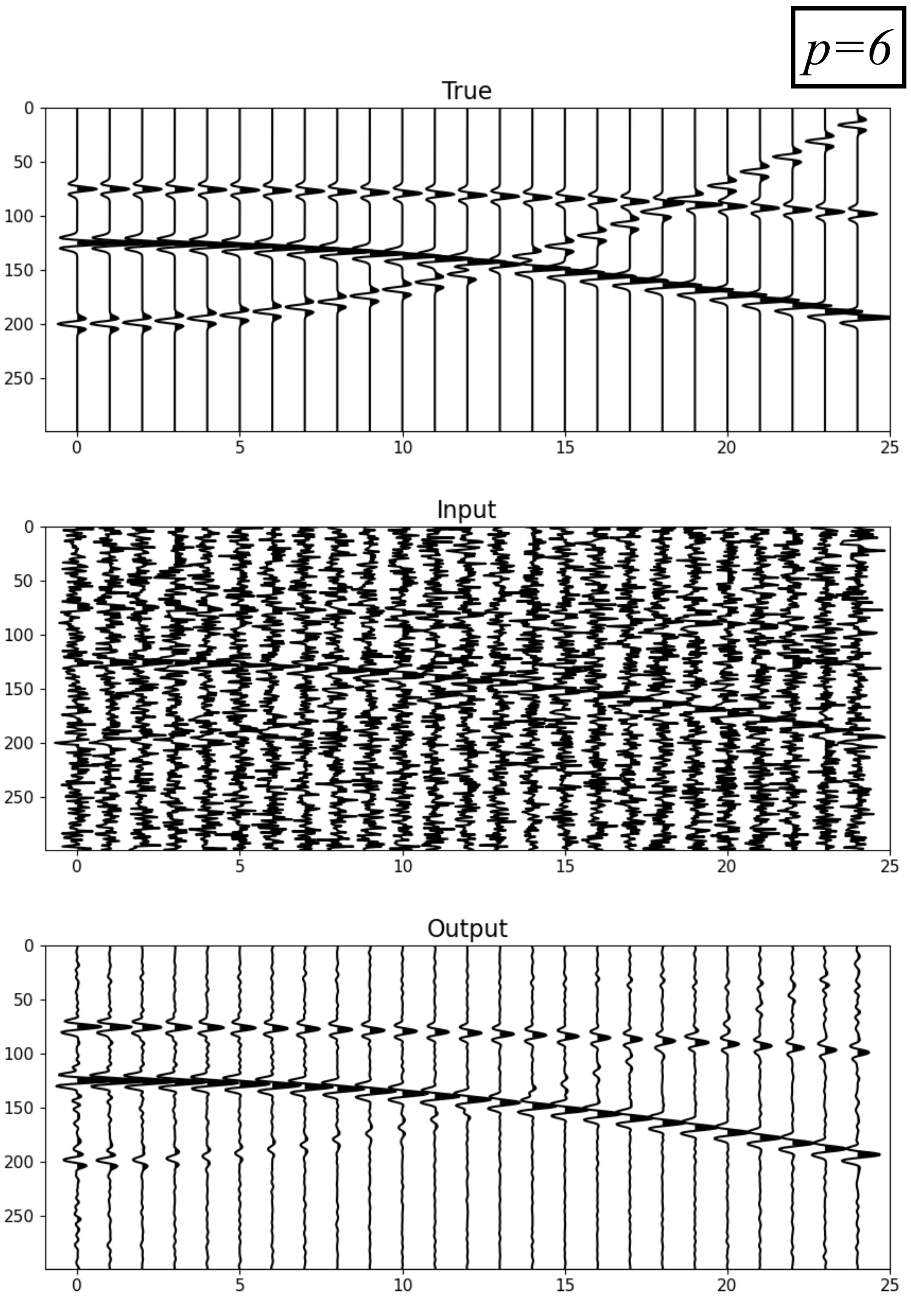
3D SSA for data in f-xy

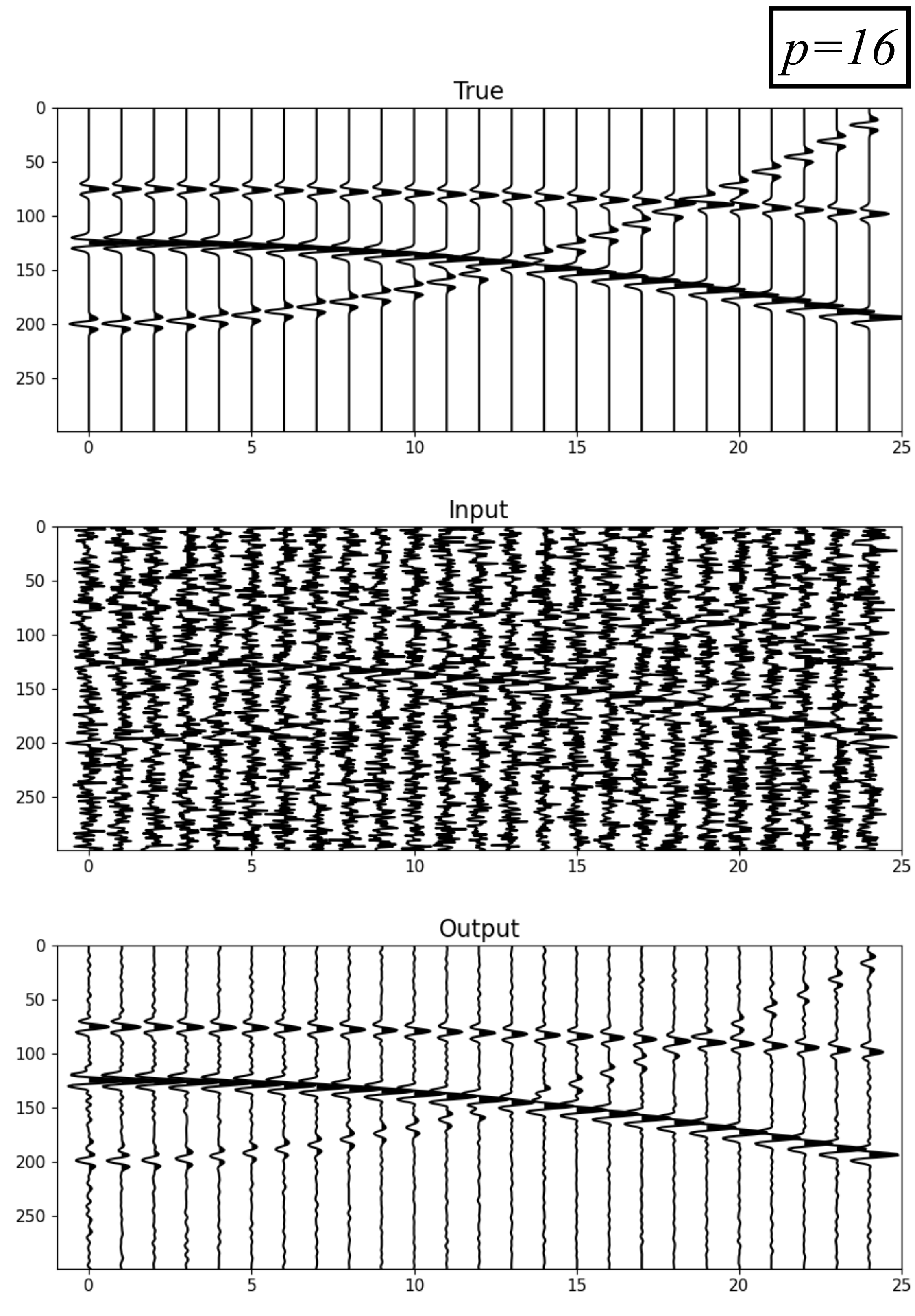


3D SSA for data in f-xy



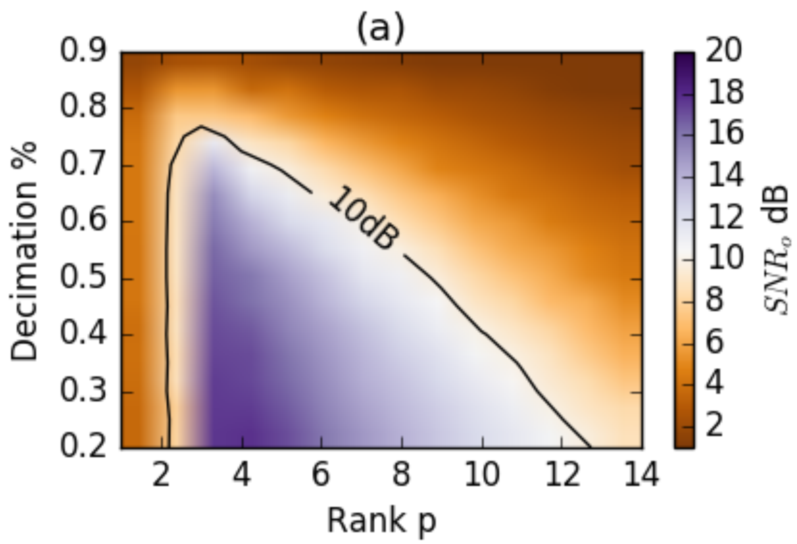
3D SSA for data in f-xy (zoom)



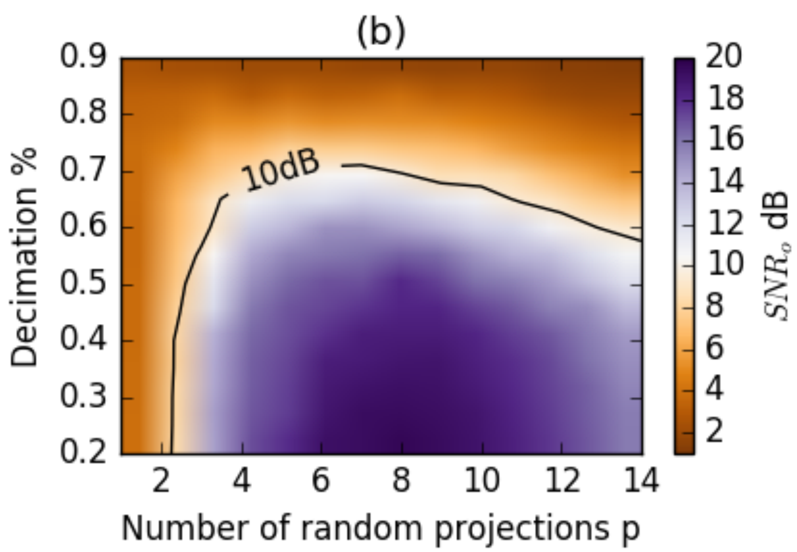


Cost

- SVD might be too expensive for the case when you need to find low-rank approximations of Block Hankel forms
- Options: Randomized QR decomposition (RQRD) or Randomized SVD or Lanczos bidiagonalization with fast matrix-times-vector multiplications
- I will use the RQRD
- *See:* N. Halko, P.-G. Martinsson, and J. A. Tropp, "Randomized Algorithms for the Low-Rank Approximation of Matrices," Proceedings of the National Academy of Sciences, vol. 108, no. 51, pp. 20491–20496, 2011
- RQRD also relaxes having to know the desired rank. See next page.



Probability of success of an algorithm used for data reconstruction using SVD



Probability of success of al algorithm used for data reconstruction using RQRD

Fernanda Carozzi, Mauricio D. Sacchi; Robust tensor-completion algorithm for 5D seismic-data reconstruction. *Geophysics* 2019;; 84 (2): V97–V109

RQRD

```
function rqrd(H, p)
    # Applying randomized QR for rank reduction

    # Get the number of columns
    ny = size(H, 2)

    # Generate a random matrix, p is like a relaxed rank
    R = randn(ny, p)

    # Compute the product
    Y = H * R

    # QR decomposition
    F = qr(Y)
    Q=F.Q

    # Calculate the output
    Hp = Q[:,1:p] * Q[:,1:p]' * H

    return Hp
end
```

Random

$$\mathbf{Y} = \mathbf{H}\mathbf{R}$$

Hankel

RQRD

In the previous slide, we need to evaluate a product of the form

$$\mathbf{Y} = \mathbf{H}\mathbf{R}$$

For each column we have $\mathbf{y} = \mathbf{A}\mathbf{r}$

- Hankel-times vector can be done via FFTs, so the p products are computed via FFTs
- Level-2 Block Hankel times vector can also via done via FFTs (2D FFTs)

Hankel and Circulant Matrices and.. FFTs!!!

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \rightarrow \mathbf{H}(\mathbf{s}) = \begin{pmatrix} s_1 & s_2 & s_3 \\ s_2 & s_3 & s_4 \\ s_3 & s_4 & s_5 \\ s_4 & s_5 & s_6 \end{pmatrix}$$

$$\begin{pmatrix} s_1 & s_2 & s_3 \\ s_2 & s_3 & s_4 \\ s_3 & s_4 & s_5 \\ s_4 & s_5 & s_6 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_3 & s_2 & s_1 \\ s_4 & s_3 & s_2 \\ s_5 & s_4 & s_3 \\ s_6 & s_5 & s_4 \end{pmatrix} = \mathbf{T}$$

Toeplitz

$$\mathbf{H\Phi} = \mathbf{T}$$

Hankel and Circulant Matrices and.. FFTs!!!

Reverse operator: $\mathbf{\Phi} = \mathbf{\Phi}^T = \mathbf{\Phi}^{-1}\,, \quad \mathbf{\Phi}\mathbf{\Phi} = \mathbf{I}$

$$\mathbf{H}\,\mathbf{m} = \begin{pmatrix} s_1 & s_2 & s_3 \\ s_2 & s_3 & s_4 \\ s_3 & s_4 & s_5 \\ s_4 & s_5 & s_6 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \mathbf{H}\,\mathbf{\Phi}\,\mathbf{\Phi}\,\mathbf{m} = \mathbf{T} \begin{pmatrix} m_3 \\ m_2 \\ m_1 \end{pmatrix}$$

To multiply a Hankel times a vector is equivalent to multiply a Toeplitz matrix times up-down flipped vector

Hankel and Circulant Matrices and.. FFTs!!!

Circulant form

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \to \mathbf{C} = \begin{pmatrix} c_1 & c_6 & c_5 & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_6 & c_5 & c_4 & c_3 \\ c_3 & c_2 & c_1 & c_6 & c_5 & c_4 \\ c_4 & c_3 & c_2 & c_1 & c_6 & c_5 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_6 \\ c_6 & c_5 & c_4 & c_3 & c_2 & c_1 \end{pmatrix}$$

Hankel and Circulant Matrices and.. FFTs!!!

Circulant form

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \rightarrow \mathbf{C} = \begin{pmatrix} c_1 & c_6 & c_5 & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_6 & c_5 & c_4 & c_3 \\ c_3 & c_2 & c_1 & c_6 & c_5 & c_4 \\ c_4 & c_3 & c_2 & c_1 & c_6 & c_5 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_6 \\ c_6 & c_5 & c_4 & c_3 & c_2 & c_1 \end{pmatrix}$$

Toeplitz

Hankel and Circulant Matrices and.. FFTs!!!

Circulant form

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \rightarrow \mathbf{C\Phi m} = \begin{pmatrix} c_1 & c_6 & c_5 & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_6 & c_5 & c_4 & c_3 \\ c_3 & c_2 & c_1 & c_6 & c_5 & c_4 \\ c_4 & c_3 & c_2 & c_1 & c_6 & c_5 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_6 \\ c_6 & c_5 & c_4 & c_3 & c_2 & c_1 \end{pmatrix} \begin{pmatrix} m_3 \\ m_2 \\ m_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

Toeplitz

Desired vector in blue

Hankel and Circulant Matrices and.. FFTs!!!

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \rightarrow \mathbf{C\Phi m} = \begin{pmatrix} c_1 & c_6 & c_5 & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_6 & c_5 & c_4 & c_3 \\ c_3 & c_2 & c_1 & c_6 & c_5 & c_4 \\ c_4 & c_3 & c_2 & c_1 & c_6 & c_5 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_6 \\ c_6 & c_5 & c_4 & c_3 & c_2 & c_1 \end{pmatrix} \begin{pmatrix} m_3 \\ m_2 \\ m_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

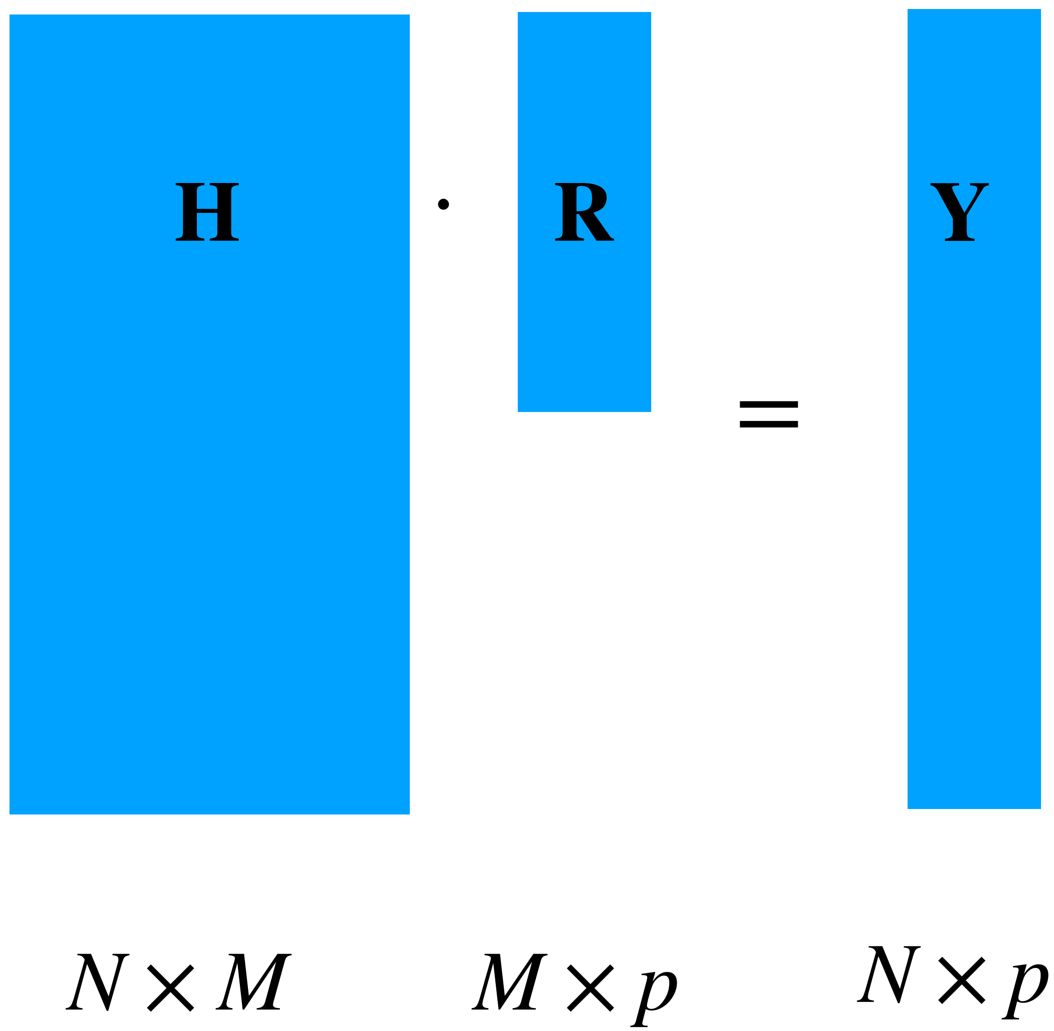
$$\textbf{ifft}\left[\textbf{fft} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{pmatrix} \circ \textbf{fft} \begin{pmatrix} m_3 \\ m_2 \\ m_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

Fast Algorithm for SSA (FSSA)

- Adopt RQRD
- Randomize Hankel matrix with the FFT trick
- Use FFT trick

Fast Algorithm for SSA (FSSA)

1)



Use fft for multiplications without forming H!

2) $\mathbf{Q}, \mathbf{R} = \mathbf{QR}[\,\mathbf{Y}\,]$

3) $\mathbf{H}_p = \mathbf{Q}\,\mathbf{Q}^H\,\mathbf{H} = \mathbf{QC}$

Use fft for multiplications again

4) $\mathbf{s}_p = \mathscr{A}[\mathbf{Q}\,\mathbf{C}]$

Use convolutions also via fft to perform Antidiagonal averaging

Jinkun Cheng, Mauricio Sacchi, Jianjun Gao; Computational efficient multidimensional singular spectrum analysis for prestack seismic data reconstruction. *Geophysics* 2019;; 84 (2): V111–V119