

# Sparsity and Fourier Reconstruction

**Mauricio D Sacchi**  
**SAIG and University of Alberta**

[msacchi@ualberta.ca](mailto:msacchi@ualberta.ca)

# Contents

●	FOURIER RECONSTRUCTION	
●	LINEAR INVERSE PROBLEMS	16
●	TWO IMPORTANT MATRICES (OPERATORS)	36
●	DISCRETE LINEAR INVERSE PROBLEMS IN EXPLICIT FORM	45
●	PROBLEMS IN IMPLICIT FORM	95

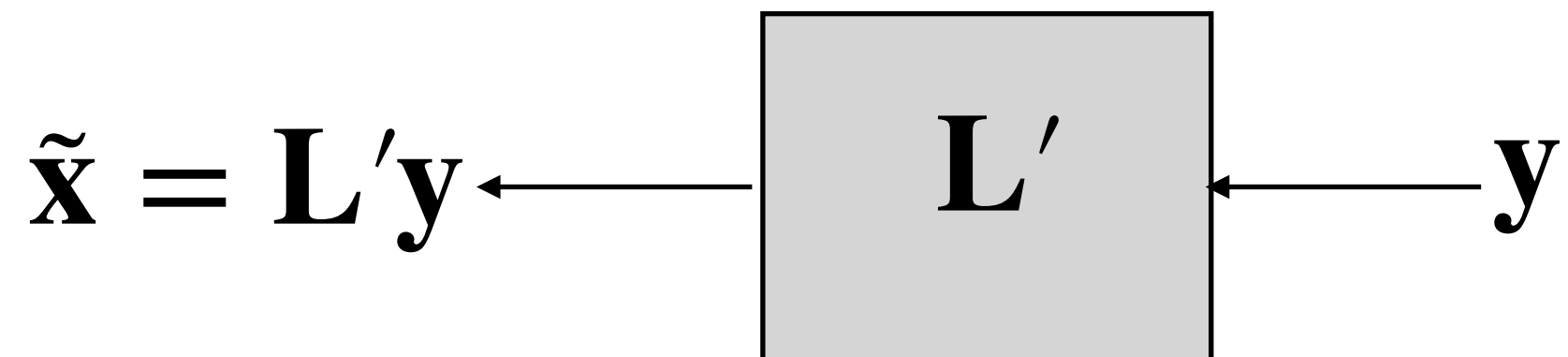
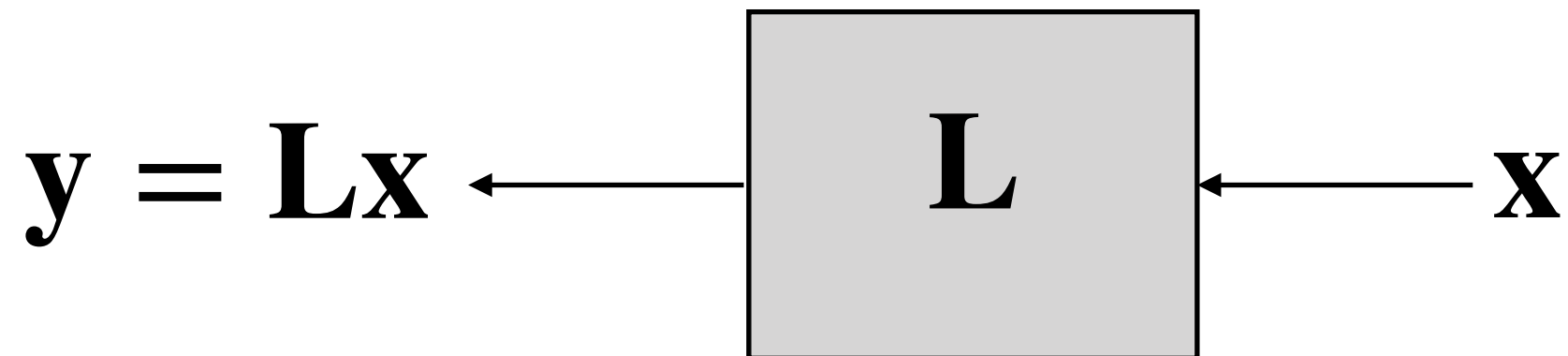
# Fourier Reconstruction

- I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm", *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600-616, Mar. 1997.
- M. D. Sacchi, T. J. Ulrych and C. J. Walker, "Interpolation and extrapolation using a high-resolution discrete Fourier transform," in *IEEE Transactions on Signal Processing*, vol. 46, no. 1, pp. 31-38, Jan. 1998, doi: 10.1109/78.651165.
- Bin Liu and M D Sacchi, "Minimum weighted norm interpolation of seismic records", *GEOPHYSICS* 2004 69:6, 1560-1568
- Daniel Trad, "Five-dimensional interpolation: Recovering from acquisition constraints" *GEOPHYSICS* 2009 74:6, V123-V132
- Paul Zwartjes and D. Gisolf, "Fourier reconstruction with sparse inversion", *Geophysical Prospecting*, 2007, 55:2, 199-221

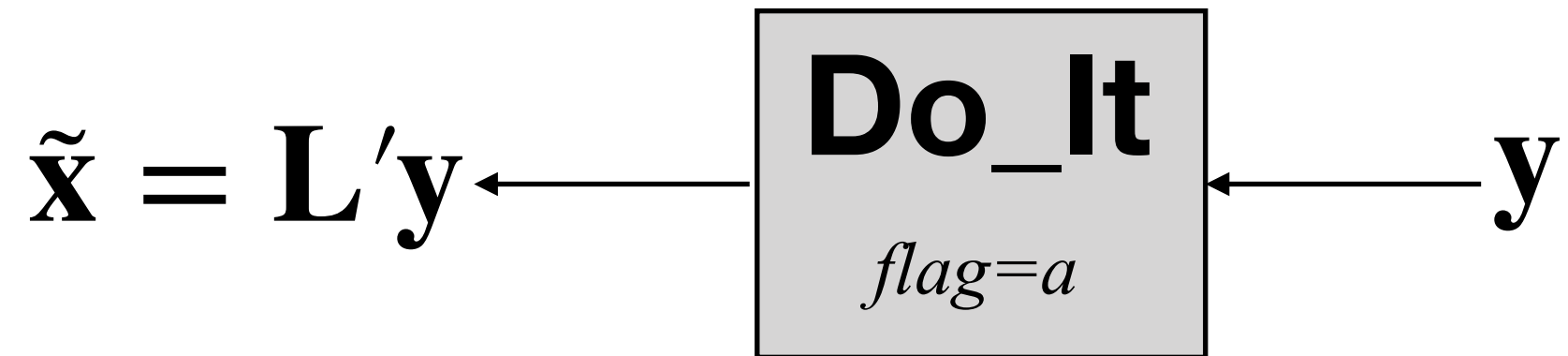
# Interpolation problem

- Data to interpolate is not sparse
- Coefficients representing the data are sparse if the basis functions to represent the data are properly selected
- Let's discuss how one can solve the interpolation problem via a **sparsity promoting** algorithm
  - First we will use **explicit** operator (Matrices)
  - Then Transforms applied as **implicit** linear operators

# Explicit



# Implicit



*f*: Forward

*a*: Adjoint or transpose

# Interpolation problem

- Given ideal data  $\mathbf{m}$
- Assume the data was sampled to produce observations  $\mathbf{d}$
- $\mathbf{d} = \mathbf{S}\mathbf{m}$
- Where  $\mathbf{S}$  is the Sampling Operator that extracts  $N$  samples of the length  $N$  signal  $\mathbf{m}$
- $\mathbf{S} \in \mathcal{R}^{N \times M}$

# Interpolation Problem: **Sampling Operator**

- Given ideal complete data  $\mathbf{m}$  and a sampling operator that produce observations  $\mathbf{d}$  which are a subset of  $\mathbf{m}$

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \\ m_9 \end{bmatrix} \xrightarrow{\text{Flavour 1 Operator}} \mathbf{d} = \mathbf{S}\mathbf{m}$$

- Where  $\mathbf{S}$  is the Sampling Operator that extracts  $N$  samples of the length  $M$  signal  $\mathbf{m}$
- $\mathbf{S} \in \mathcal{R}^{N \times M}$ . The goal is to estimate  $\mathbf{m}$  from  $\mathbf{d}$  (underdetermined)



# Interpolation Problem: Sampling Operator

- The adjoint of sampling replaces empty bins by zeros

$$\begin{bmatrix} d_1 \\ 0 \\ 0 \\ d_2 \\ 0 \\ d_3 \\ 0 \\ 0 \\ d_4 \end{bmatrix} = \mathbf{S}^T \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} \longrightarrow \tilde{\mathbf{m}} = \mathbf{S}^T \mathbf{d}$$

Properties that can be easily shown:  $\mathbf{S}\mathbf{S}^T = \mathbf{I}$ ,  $\mathbf{S}^T\mathbf{S} \neq \mathbf{I}$

# Interpolation Problem: Naive Solution

Because is an underdetermined problem, we could try the minimum norm solution:

$$\mathbf{d} = \mathbf{S}\mathbf{m} \longrightarrow \mathbf{m}_{mn} = \mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{d}$$

$$\text{But } \mathbf{S}\mathbf{S}^T = \mathbf{I} \longrightarrow \mathbf{m}_{mn} = \mathbf{S}^T\mathbf{d}$$

$$(1) \quad \mathbf{m}_{mn} = \mathbf{S}^T \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ 0 \\ 0 \\ d_2 \\ 0 \\ d_3 \\ 0 \\ 0 \\ d_4 \end{bmatrix} = \begin{bmatrix} m_1 \\ 0 \\ 0 \\ m_4 \\ 0 \\ m_6 \\ 0 \\ 0 \\ m_9 \end{bmatrix}$$

The interpolation via the minimum norm solution did not work. There is nothing better than zeros to make the norm of the solution small!

# Interpolation Problem: Naive Solution, second attempt

Because is an underdetermined problem, we could try the minimum norm solution but on an orthogonal basis like Fourier.

$$\mathbf{d} = \mathbf{S}\mathbf{m}$$

$$\mathbf{m} = \mathbf{F}\mathbf{a} \quad \text{The Fourier inverse transform is } \mathbf{F} \text{ with } \mathbf{F}\mathbf{F}^H = \mathbf{I}$$

$$\mathbf{d} = \mathbf{S}\mathbf{F}\mathbf{a} \quad \text{Then,} \quad \mathbf{a}_{mn} = \mathbf{F}^H \mathbf{S}^T (\mathbf{S}\mathbf{F}\mathbf{F}^H \mathbf{S}^T)^{-1} \mathbf{d} = \mathbf{F}^H \mathbf{S}^T \mathbf{d}$$

$$\mathbf{m}_{mn} = \mathbf{F}\mathbf{a}_{mn} = \mathbf{F}\mathbf{F}^H \mathbf{S}^T \mathbf{d} = \mathbf{S}^T \mathbf{d}$$

See (1) in  
previous slide

Upppps!! again, we have replaced missing data by  
zeros like in the previous example

# Interpolation: Sparse or Compressive Solution


- Minimum norm solution in data space or transformed domain space did not work!! So we need something else. What about using sparsity?
- Asking for the solution to be sparse does not make any sense. However, we can say that solution can be represented by a transform (Forward or Synthesis transform)

$$\mathbf{m} = \mathbf{F}\mathbf{a}$$

- Where  $\mathbf{F}$  is the forward transform and  $\mathbf{a}$  are the coefficients that model the ideal data  $\mathbf{m}$
- So we have the following two problems:

# Interpolation: Sparse or Compressive Solution

- Find the coefficient such that

$$\mathbf{d} = \mathbf{S}\mathbf{m}, \quad \mathbf{m} = \mathbf{F}\mathbf{a},$$


Measurements

- The above is equivalent to  $\mathbf{d} = \mathbf{S}\mathbf{F}\mathbf{a}$ ,
- Which can be solved by minimizing the l2-l1 cost function

$$J = \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu \|\mathbf{a}\|_1$$

# Interpolation

- Interpolated data is found by solving

$$\hat{\mathbf{a}} = \operatorname{argmin}_{\mathbf{a}} \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu \|\mathbf{a}\|_1$$

$$\hat{\mathbf{m}} = \mathbf{F}\hat{\mathbf{a}},$$

- This is also the main idea behind CS (Compressive Sensing) where Sampling is Random
- Can be applied to ND-problems by using an ND transform (e.g. ND Fourier or Curvelet transforms)

# Interpolation: Example (Explicit)

- Interpolated data is found by solving

$$\hat{\mathbf{a}} = \operatorname{argmin}_{\mathbf{a}} \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu \|\mathbf{a}\|_1$$

$$\hat{\mathbf{m}} = \mathbf{F}\hat{\mathbf{a}},$$

- I can write the transform in matrix form for some simple problems. For instance the Fourier synthesis operator can be written easily as a matrix (DFT Matrix)
- Of course, it is much faster to use the FFT rather than the DFT expressed as a matrix multiplication operation

# Interpolation: Example (Explicit)

- F is the inverse DFT, that we will call the Fourier Synthesis Operator

$$x_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X_n e^{i2\pi nk/N}$$

- In matrix form

$$\mathbf{x} = \mathbf{F} \mathbf{X} \qquad \mathbf{X} = \mathbf{F}^H \mathbf{x}$$

- Where the elements of the Fourier operator (Matrix) are

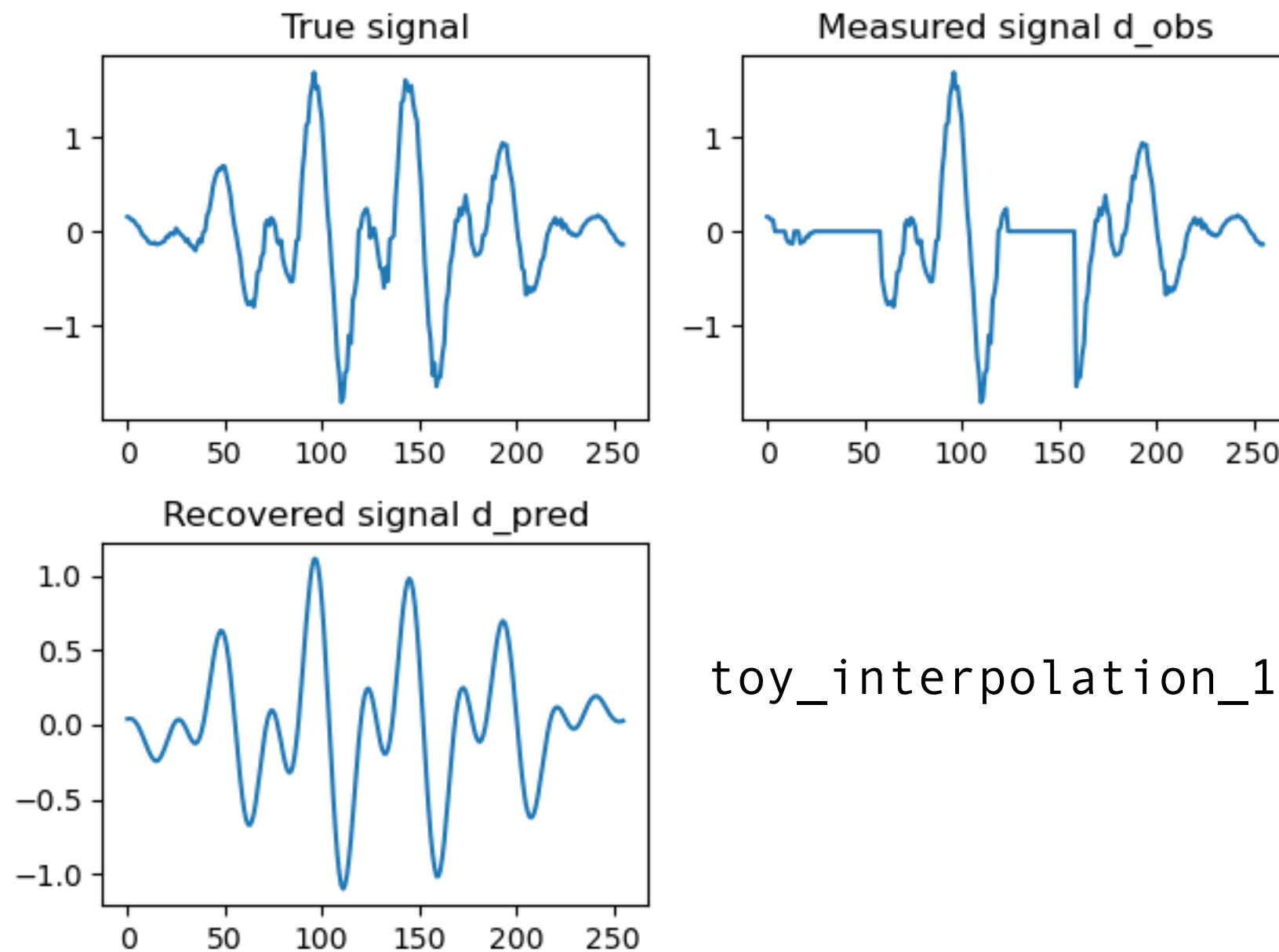
$$F_{n,k} = \frac{1}{\sqrt{N}} e^{i2\pi nk/N}$$



## Explicit DFT-matrix IRLS code

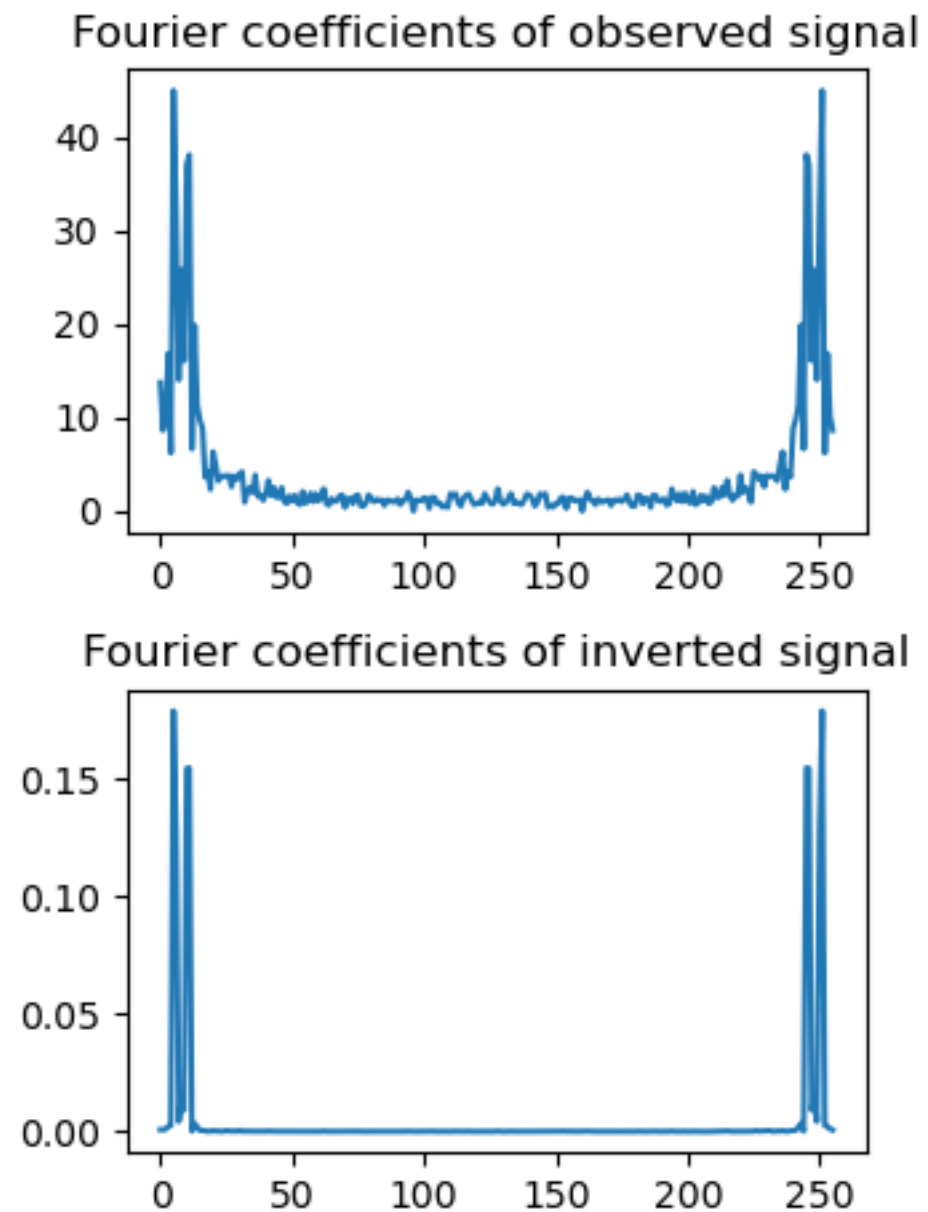
```
L = S*F          # This is the linear operator to invert
mu = 10.1
# Start IRLS
a = L'*d_obs
for k=1:10
    q = 1.0./(abs.(a).+0.001)
    Q = Diagonal(q)
    a = (L'*L+mu*Q)\(L'*d_obs)
end
d_interp = F*a
```

# Interpolation: Example (Explicit)



toy\_interpolation\_1.ipynb

# Interpolation: Example (Explicit)



toy\_interpolation\_1.ipynb

**This examples shows the initial and inverted (Sparse) Fourier Coefficients**

# Interpolation: Example (Implicit)

- Interpolated data is found by solving

$$\hat{\mathbf{a}} = \operatorname{argmin}_{\mathbf{a}} \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu \|\mathbf{a}\|^2$$

$$\hat{\mathbf{m}} = \mathbf{F}\hat{\mathbf{a}},$$

- Now in the implicit case,  $\mathbf{F}$  is not a matrix,  $\mathbf{F} = \mathbf{F}\mathbf{F}^T$
- Therefore, I can't treat the problem as before and construct operators such as  $((\mathbf{S}\mathbf{F})^H \mathbf{S}\mathbf{F} + \mu \mathbf{Q})^{-1}$  as needed by the sparse inversion solver (IRLS).

# Interpolation: Example (Implicit)

- Let's review IRLS

$$\hat{\mathbf{a}} = \operatorname{argmin}_{\mathbf{a}} \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu \|\mathbf{a}\|_1$$

- Gradient of cost  $\mathbf{g} = \mathbf{L}^H(\mathbf{L}\mathbf{a} - \mathbf{d}) + \mu\mathbf{Q}\mathbf{a}$
- With  $\mathbf{L} = \mathbf{S}\mathbf{F}$   $\mathbf{Q}_{ii} = 1/(|a|_i + \epsilon)$
- Steepest descent iteration

$$\mathbf{a}^{new} = \mathbf{a}^{old} - \lambda[\mathbf{L}^H(\mathbf{L}\mathbf{a}^{old} - \mathbf{d}) + \mu\mathbf{Q}\mathbf{a}^{old}]$$

# Interpolation: Example (Implicit)

- Steepest descent iteration

$$\mathbf{a}^{new} = \mathbf{a}^{old} - \lambda[\mathbf{L}^H(\mathbf{L}\mathbf{a}^{old} - \mathbf{d}) + \mu\mathbf{Q}\mathbf{a}^{old}]$$

- Replace explicit operators (Matrices) by implicit FFTs

$$\mathbf{L}[\cdot] = \mathbf{S}[\text{ifft}[\cdot]]$$

$$\mathbf{L}^H[\cdot] = \mathbf{fft}[\mathbf{S}^T[\cdot]]$$

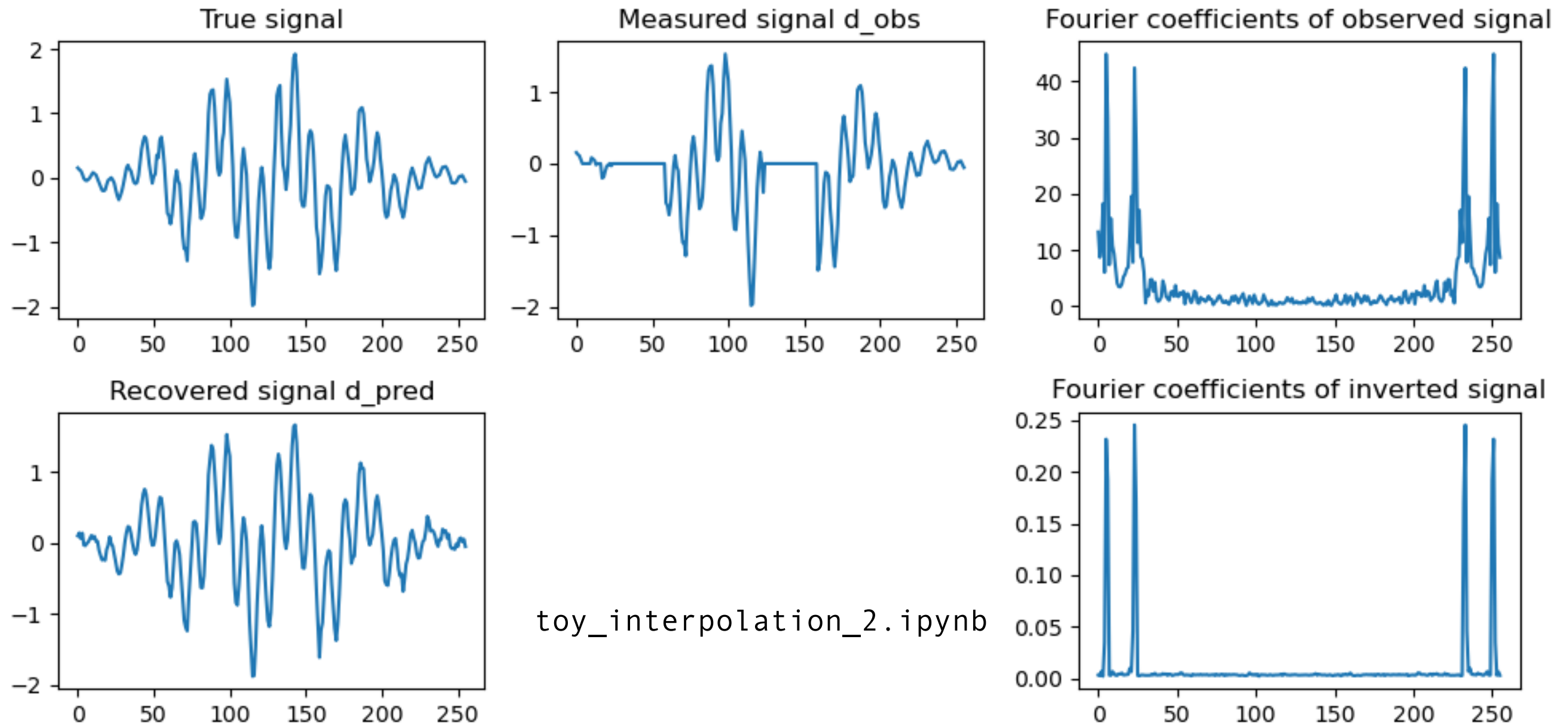
- Program does not require Fourier DFT matrix and hence is faster!!

## Implicit FFT code via steepest descent

```
for k=1:400
    q = 1.0./(abs.(a).+0.01)
    Q = Diagonal(q)
    r = S*(bfft(a))-d_obs          # r = SFa-d_obs
    a = a .-0.005*(fft(S'*r).+ mu*Q*a)  # a = a -0.005*(F^H S^T(r))
    Save_P[k,:]=abs.(a)
    Save_Q[k,:]=abs.(q)
end
```

Be careful with the normalization of the FFT and IFFT because I really need the adjoint not the inverse transform (above I use bfft)

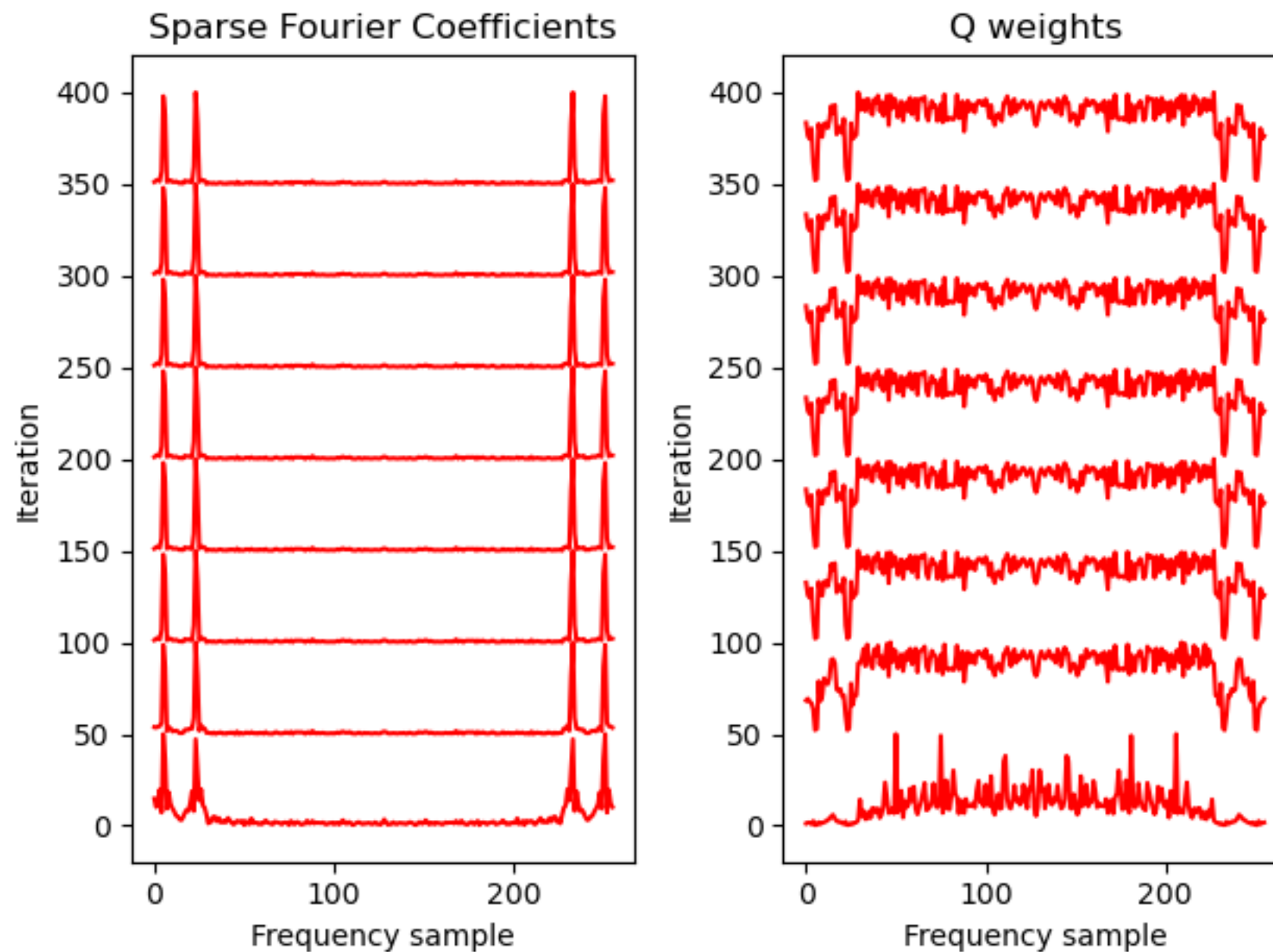
# Interpolation: Example (Implicit)



**Here I used the FFT to synthesize forward and adjoint operators**



# Interpolation: Example (Implicit)



`toy_interpolation_2.ipynb`

**Evolution of Fourier Coefficients and Q weights vs iteration**

# Two flavours of sampling operator

## Flavour 1 (matrix mult)

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \\ m_9 \end{bmatrix}$$

$$\mathbf{d} = \mathbf{S}\mathbf{m}$$

$$J = \|\mathbf{d} - \mathbf{S}\mathbf{F}\mathbf{a}\|_2^2 + \mu\|\mathbf{a}\|_1$$

## Flavour 2 (element-to-element mult)

$$\begin{bmatrix} d_1 \\ 0 \\ 0 \\ d_2 \\ 0 \\ d_3 \\ 0 \\ 0 \\ d_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \\ m_9 \end{bmatrix}$$

$$\mathbf{d} = \mathbf{S}\mathbf{m} = \mathbf{s} \circ \mathbf{m}$$

$$\mathbf{d} = \mathbf{s} \circ \mathbf{d} \quad \uparrow$$

Element-to-element  
multiplication

$\mathbf{s} =$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J = \|\mathbf{s} \circ [\mathbf{d} - \mathbf{F}\mathbf{a}]\|_2^2 + \mu\|\mathbf{a}\|_1$$

$$J = \|\mathbf{d} - \mathbf{s} \circ \mathbf{F}\mathbf{a}\|_2^2 + \mu\|\mathbf{a}\|_1$$