UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

CS598PS – Machine Learning for Signal Processing

Sparsity, Compressive Sensing and Random Projections

3 November 2017

Today's lecture

- Sparsity
- Compressive sensing
- Quantization, randomness and high dimensions

What is sparsity?

- Depends who you ask
- Basic idea: We want most numbers in a collection to be zero
- Too many ways to express that

A starting point

- Linear equation with multiple solutions:

$$y = \mathbf{a} \cdot \mathbf{x} \Rightarrow 2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Which solution would you pick?

$$\mathbf{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

A sparse answer

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

What MATLAB gives

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

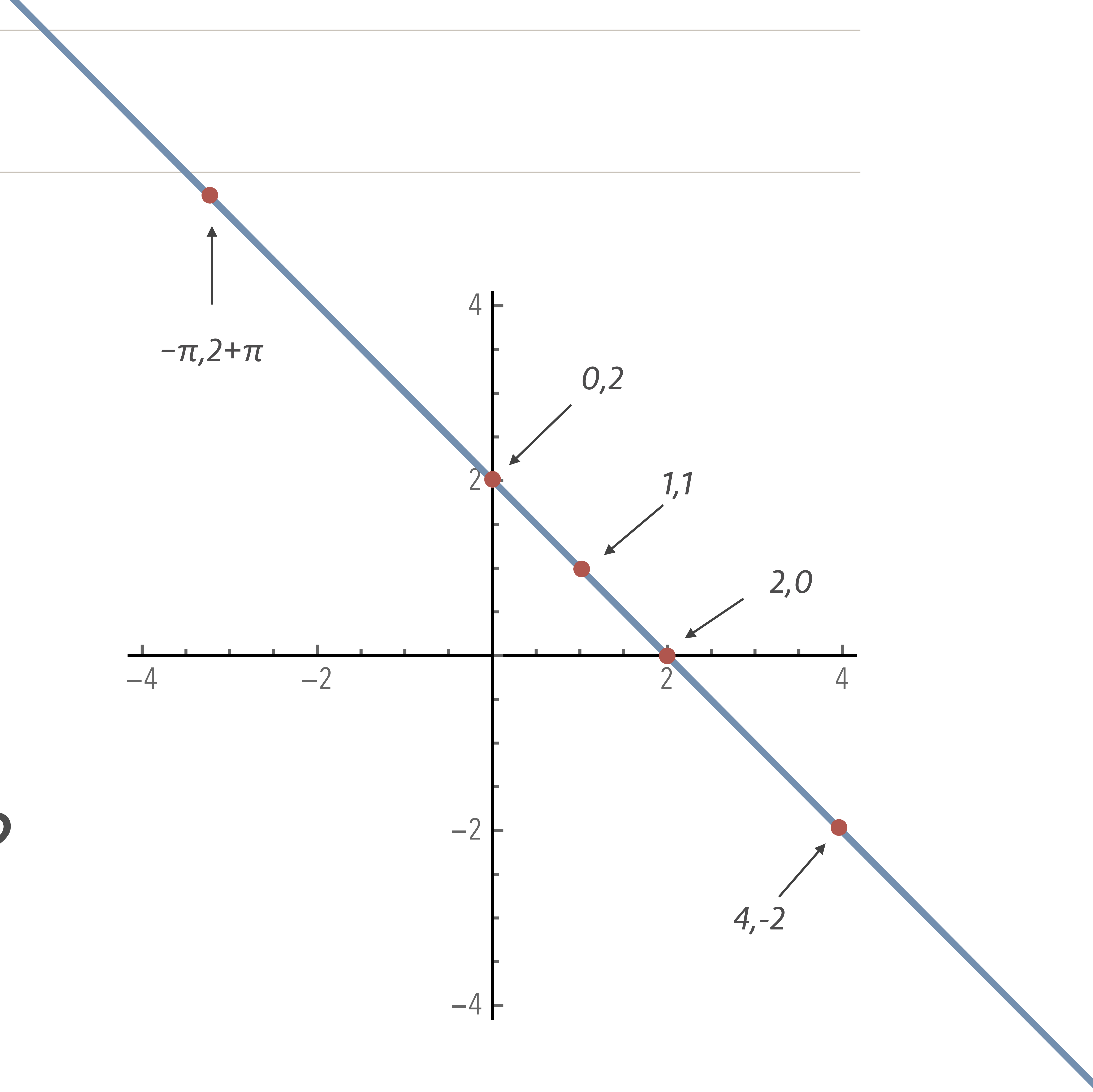
This one is fine too!

Infinite solutions

- All solutions lie on a line

- Which one do we pick?
  - Why did MATLAB pick [1, 1]?

- Does it make a difference?



The generic answer

- Least squares problem:

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x} \Rightarrow \mathbf{x} = \mathbf{A}^{+} \cdot \mathbf{y}$$
$$\Rightarrow \arg\min_{\mathbf{x}} \left(\left\| \mathbf{A} \cdot \mathbf{x} - \mathbf{y} \right\|_2 + \left\| \mathbf{x} \right\|_2 \right)$$

- Find the minimum-norm $\mathbf{x}$ that minimizes the error
  - But why $\|\mathbf{x}\|_2$?

Least squares, pseudoinverse, and ℓ_2

- Use Langrangian multipliers:

$$\left\|\mathbf{x}\right\|_2^2 + \lambda^\top \cdot \left(\mathbf{A} \cdot \mathbf{x} - \mathbf{y}\right) \Rightarrow \hat{\mathbf{x}} = -\tfrac{1}{2}\mathbf{A}^\top \cdot \lambda$$

- Put back in original equation:

$$\mathbf{A} \cdot \hat{\mathbf{x}} = -\tfrac{1}{2}\mathbf{A} \cdot \mathbf{A}^\top \cdot \lambda = \mathbf{y} \Rightarrow \lambda = -2\left(\mathbf{A} \cdot \mathbf{A}^\top\right)^{-1} \cdot \mathbf{y}$$

$$\Rightarrow \hat{\mathbf{x}} = \mathbf{A}^\top \cdot \left(\mathbf{A} \cdot \mathbf{A}^\top\right)^{-1} \cdot \mathbf{y} = \mathbf{A}^+ \cdot \mathbf{y}$$

Many more norms

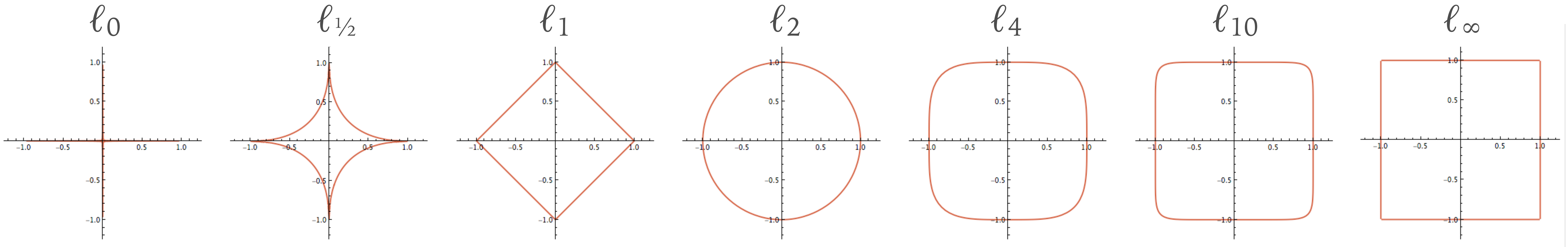
- p-Norm (or Lp / L_p / ℓ_p)

$$\left\|\mathbf{x}\right\|_p = \sqrt[p]{\sum_i \left|x_i\right|^p}$$

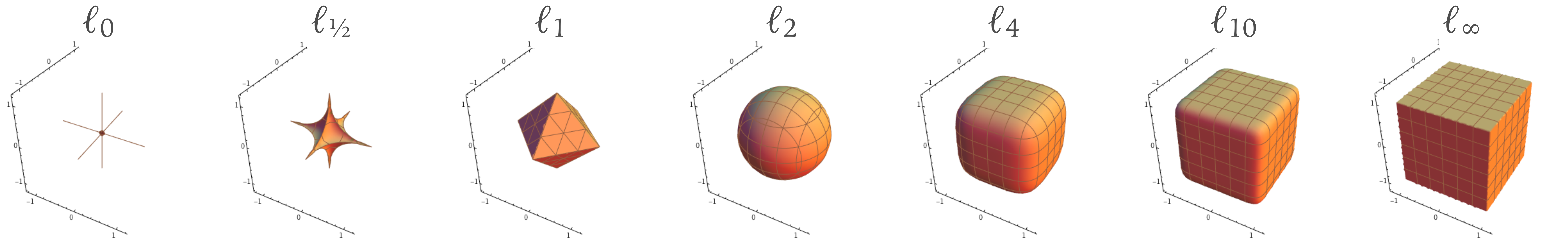
  - ℓ_2 norm is the Euclidean norm
  - ℓ_1 norm is sum of absolute values
  - ℓ_0 norm is the number of non-zero values
  - ℓ_∞ norm is max of all values

How do they look?

- Unit norms in 2D

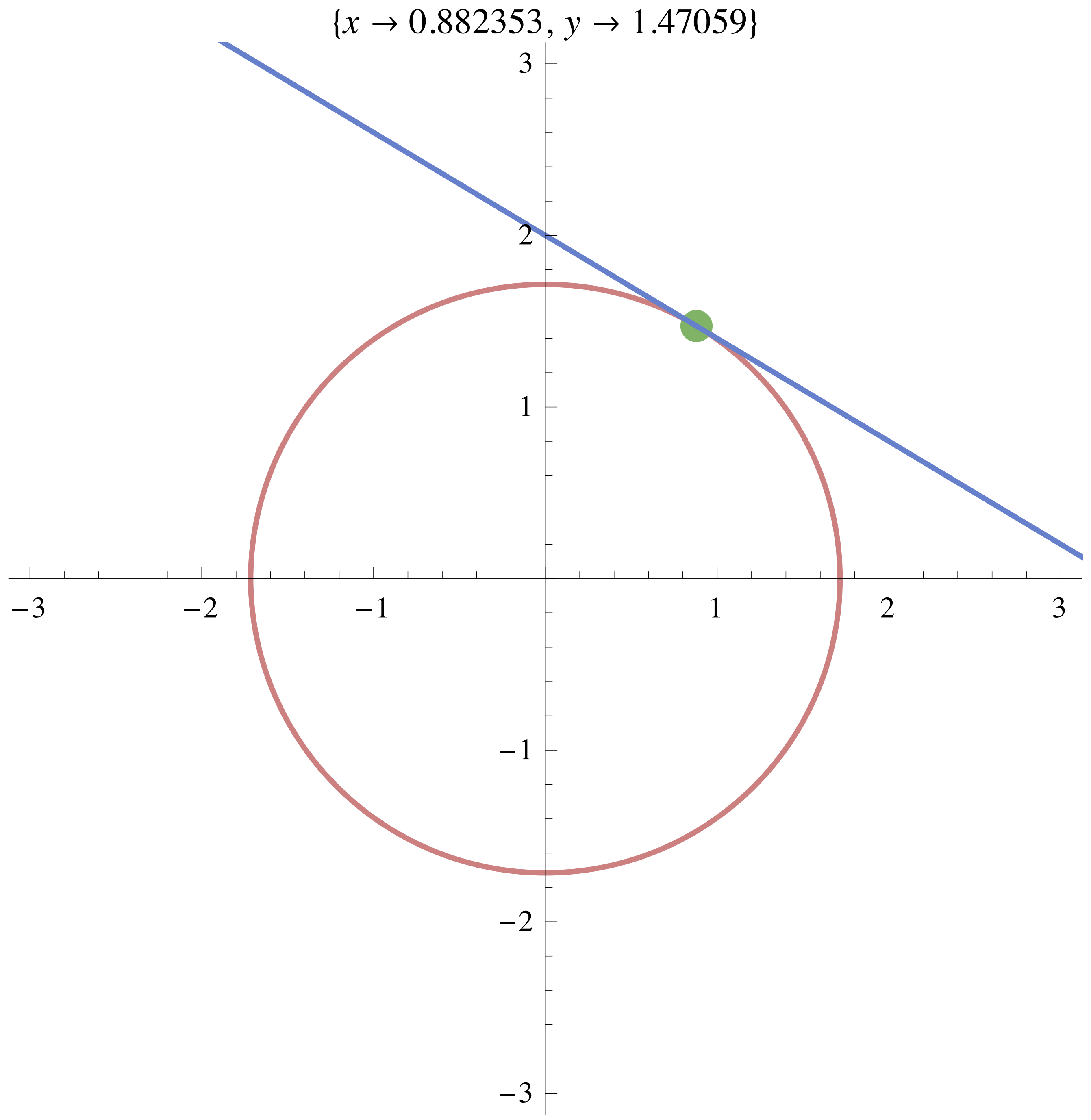


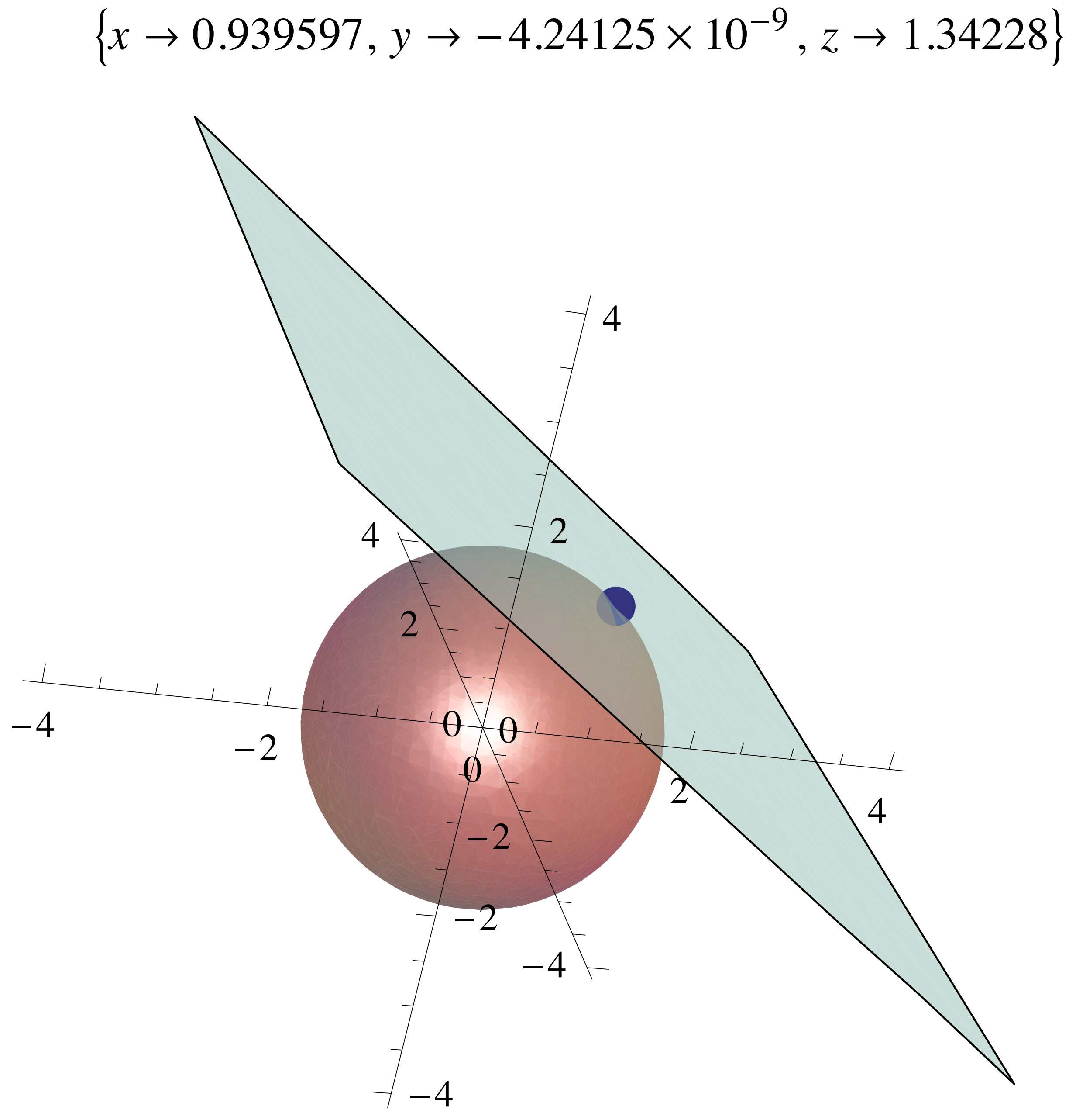
- Unit norms in 3D



ℓ_2-based pseudoinverse

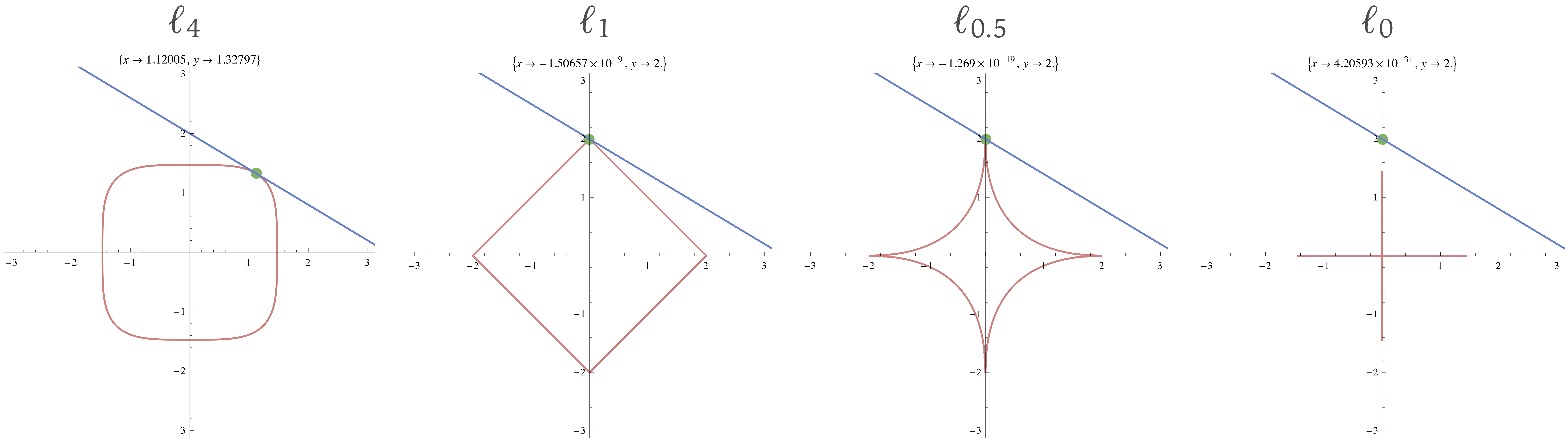
- All possible solutions lie on a hyperplane
  - Minimum ℓ_2 solution will be the point where the smallest possible ℓ_2-ball touches the solutions hyperplane





Other ℓ_p-norm solutions

- With different norms the solution will change
  - Larger p will produce a "busier" solution
  - Smaller p will produce sparser solution



Which one to use?

- For sparsity we ideally we want minimum ℓ_0
  - Directly results in smallest number of non-zero values
- But, this is an inconvenient form …

$$\ell_0(\mathbf{x}) = \sum_i [x_i \neq 0]$$

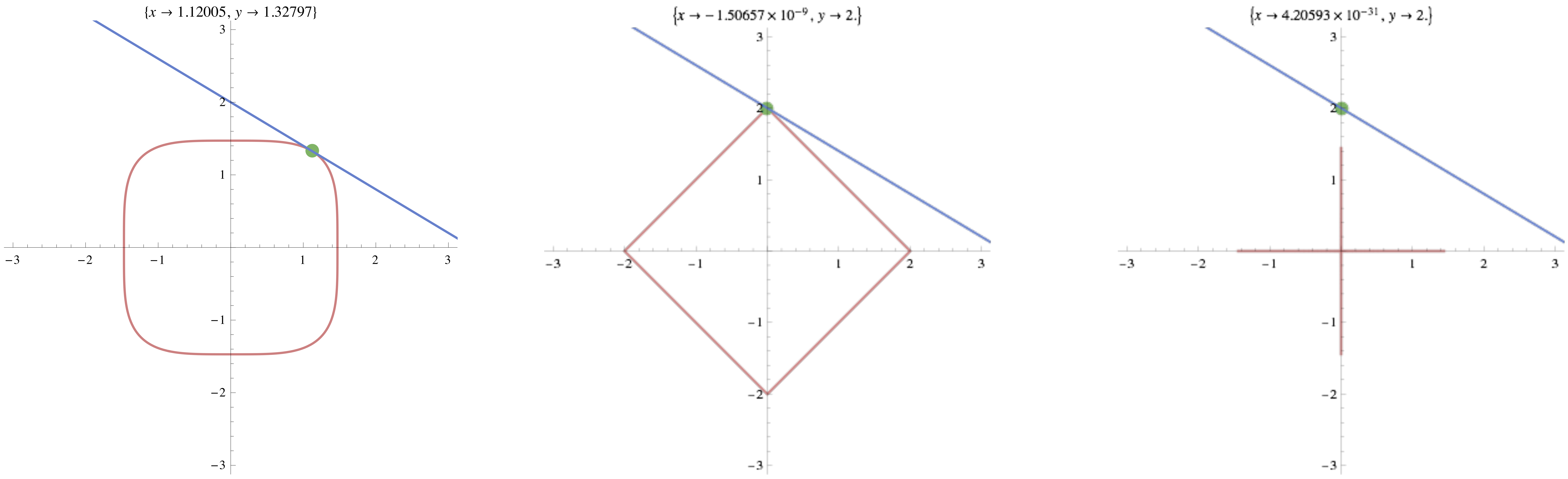
Iverson bracket:
if content evaluates to true it returns 1 otherwise it returns 0

  - Discontinuous, no derivative, not convex, etc …

$$\frac{\partial \|\mathbf{x}\|_0}{\mathbf{x}} = ? \longrightarrow \boxed{\|\mathbf{x}\|_0} + \lambda^\top \cdot (\mathbf{A}\cdot\mathbf{x} - \mathbf{y})$$

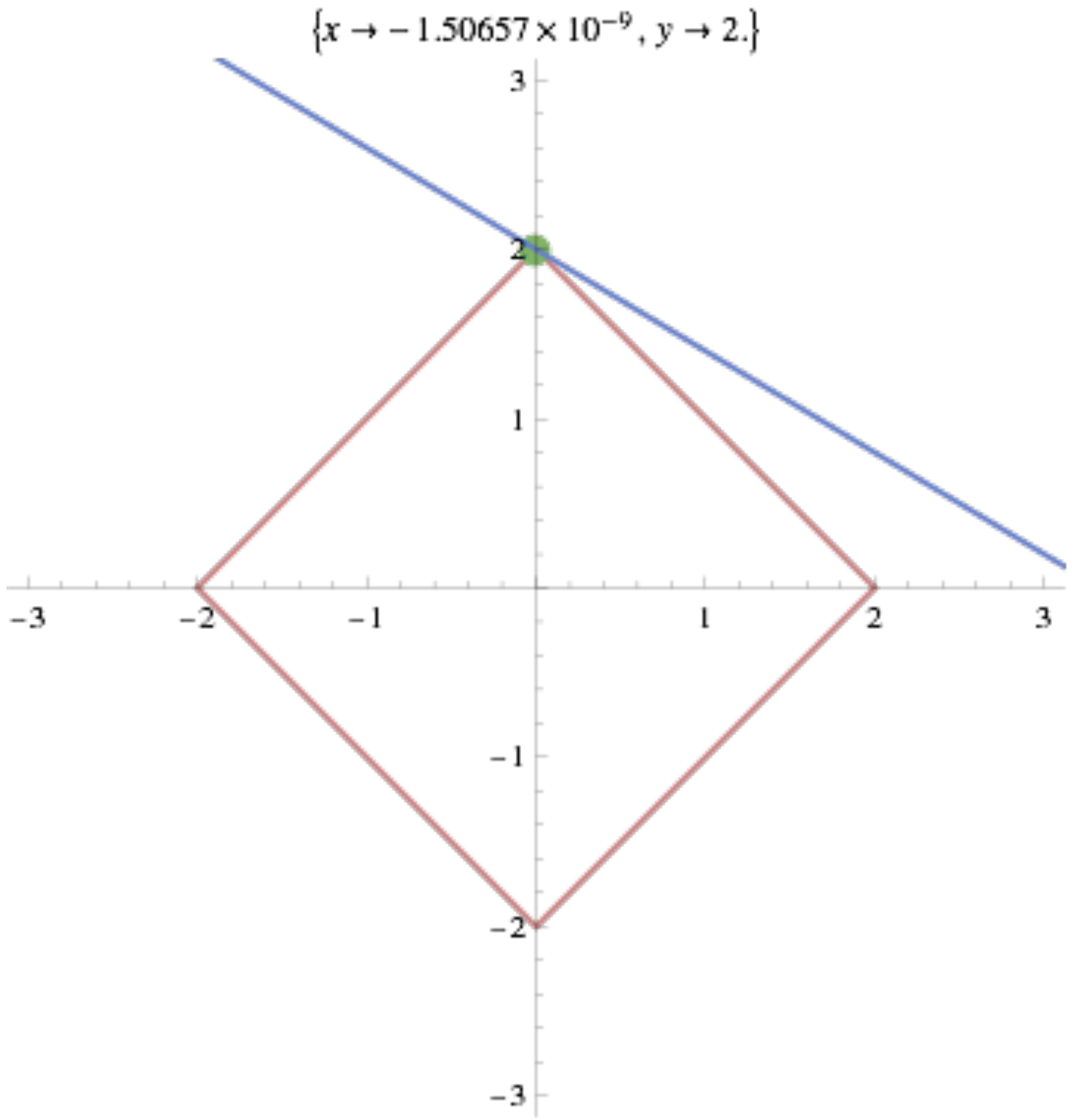
Let's try something simpler then

- How about using the ℓ_1 instead?
  - Seems to produce the same solution



Why does this work?

- The ℓ_1 case is (sort of) convex
  - ℓ_1 ball shrinks as we move towards the ideal sparse solution
  - There's one ill-defined scenario, but it isn't a big problem
    - Which is it?
- So let's solve that instead



The problem to solve

- We now have:

$$\arg\min_{\mathbf{x}} \left(\left\| \mathbf{A} \cdot \mathbf{x} - \mathbf{y} \right\|_2 + \left\| \mathbf{x} \right\|_1 \right)$$

- Minor glitch: We can't differentiate the absolute values in the ℓ_1 norm!

$$\sum_i \left| x_i \right| + \lambda^\top \cdot \left(\mathbf{A} \cdot \mathbf{x} - \mathbf{y} \right)$$

- But we can use other tools

Linear programming

- A linear program is defined as:

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \cdot \mathbf{x} \\ \text{subject to} & \mathbf{A} \cdot \mathbf{x} \leq \mathbf{y} \\ \text{and} & \mathbf{x} \geq 0 \end{array}$$

- A Nobel-prize staple of optimization theory, resource allocation, economics, etc.

Doesn't exactly match the ℓ_1 problem

- We would like to change our problem definition to fit the linear programming formulation

What we have

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & \mathbf{A} \cdot \mathbf{x} = \mathbf{y} \end{aligned}$$

What we can solve

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^\top \cdot \mathbf{x} \\ \text{subject to} \quad & \mathbf{A} \cdot \mathbf{x} \leq \mathbf{y} \\ \text{and} \quad & \mathbf{x} \geq 0 \end{aligned}$$

With some shuffling around

- We rewrite the unknown vector $\mathbf{x}$ as a difference of positive-valued vectors:

$$\mathbf{x} = \mathbf{u} - \mathbf{v}, \quad u_i, v_i \geq 0, \quad \mathbf{z} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

- Now our problem can written as:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{y} \Rightarrow \begin{bmatrix} \mathbf{A}, & -\mathbf{A} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \mathbf{y} \Rightarrow \begin{bmatrix} \mathbf{A}, & -\mathbf{A} \end{bmatrix} \cdot \mathbf{z} = \mathbf{y}$$

Now it is a linear program

- And we can solve our problem

Minimum ℓ_1 problem

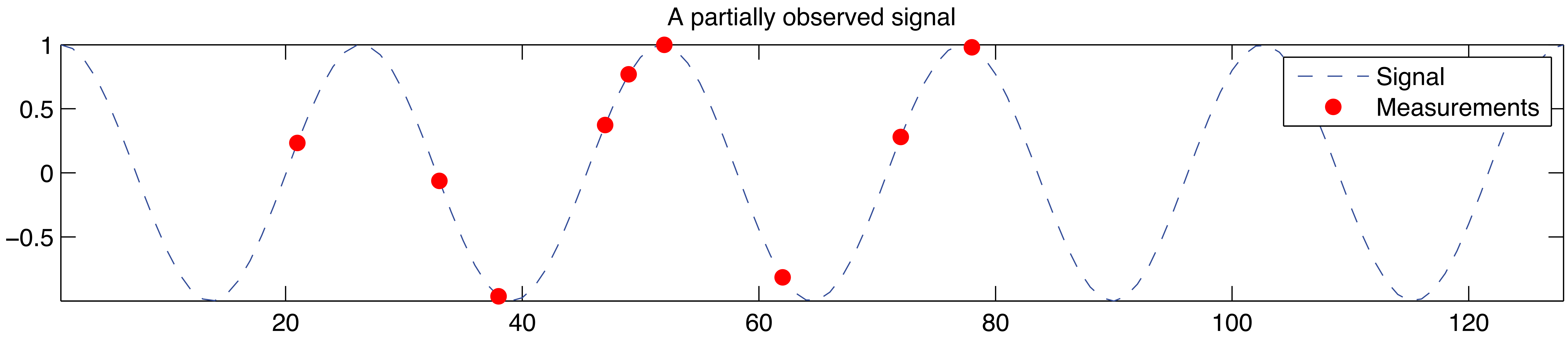
$$\begin{aligned} &\text{minimize} && \|\mathbf{x}\|_1 \\ &\text{subject to} && \mathbf{A}\cdot\mathbf{x} = \mathbf{y} \end{aligned}$$

Equivalent linear program

$$\begin{aligned} &\text{minimize} && \|\mathbf{z}\|_1 = 1^\top \cdot \mathbf{z} \\ &\text{subject to} && [\mathbf{A}, -\mathbf{A}]\cdot\mathbf{z} = \mathbf{y} \\ &\text{and} && \mathbf{z} \geq 0 \end{aligned}$$

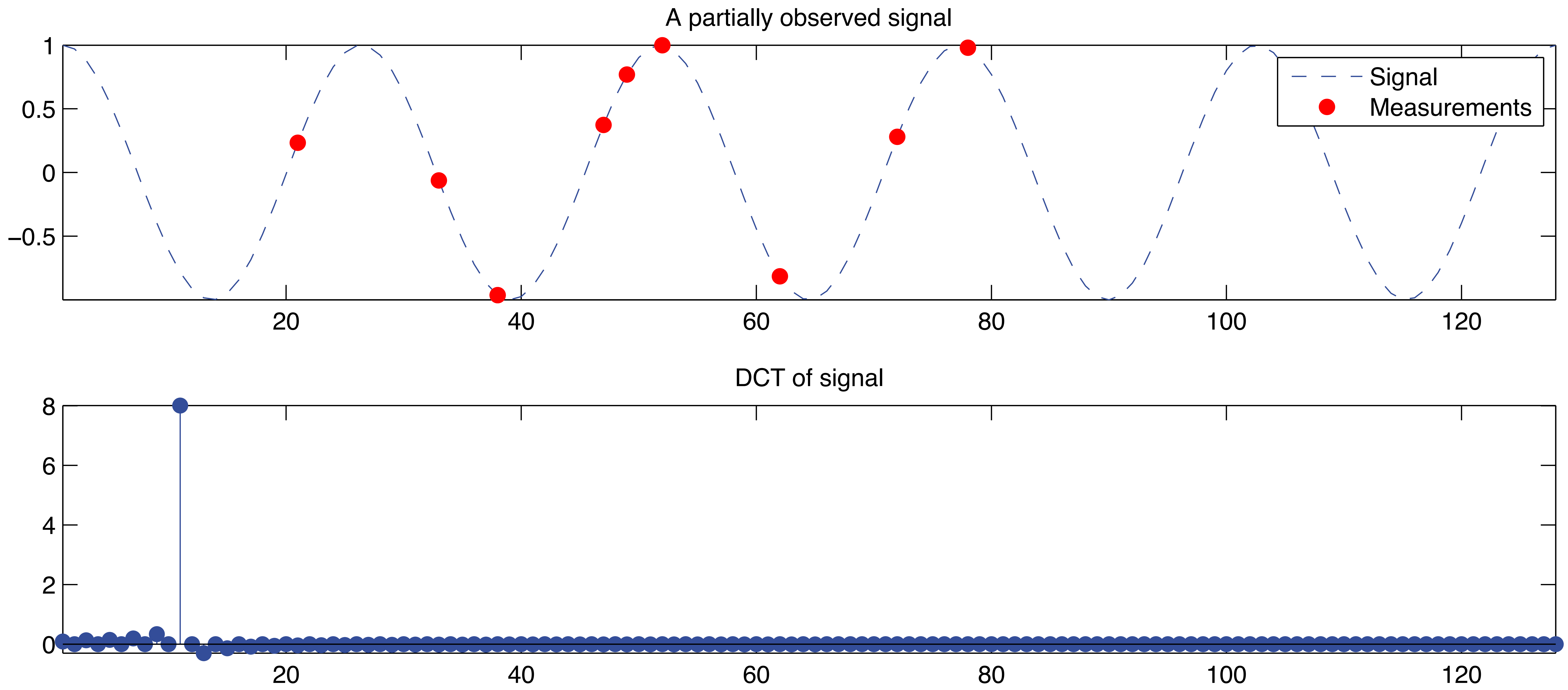
A simple example

- Suppose you measure this sinusoid
  - Can you recover the original signal using only the small number of measurements that we made?



Key observation

- The original signal is sparse in the frequency domain
  - How about we use that to construct a problem?



The problem to solve

- Find a set of small coefficients $\mathbf{x}$ in the DCT domain, and make sure that they explain all our data $\mathbf{y}$, i.e.:

$$\mathbf{A} \cdot \mathbf{C}^{-1} \cdot \mathbf{x} = \mathbf{y}$$

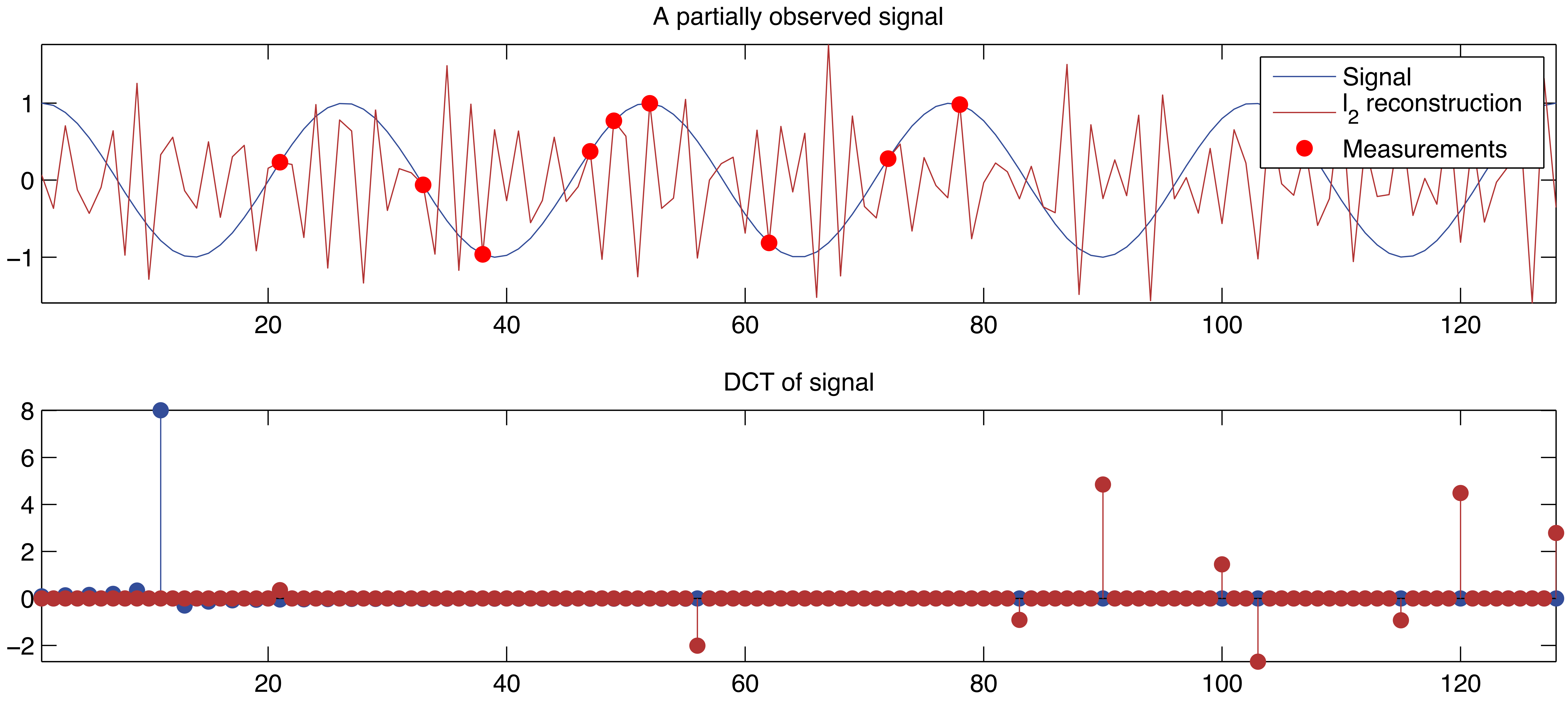
  - Where matrix $\mathbf{A}$ selects only the indices that we observe

- Simple least squares problem using minimum ℓ_2 $\mathbf{x}$

$$\mathbf{x} = \left(\mathbf{A} \cdot \mathbf{C}^{-1}\right)^{+} \cdot \mathbf{y}$$

And it doesn't really do much

- But you expected that!

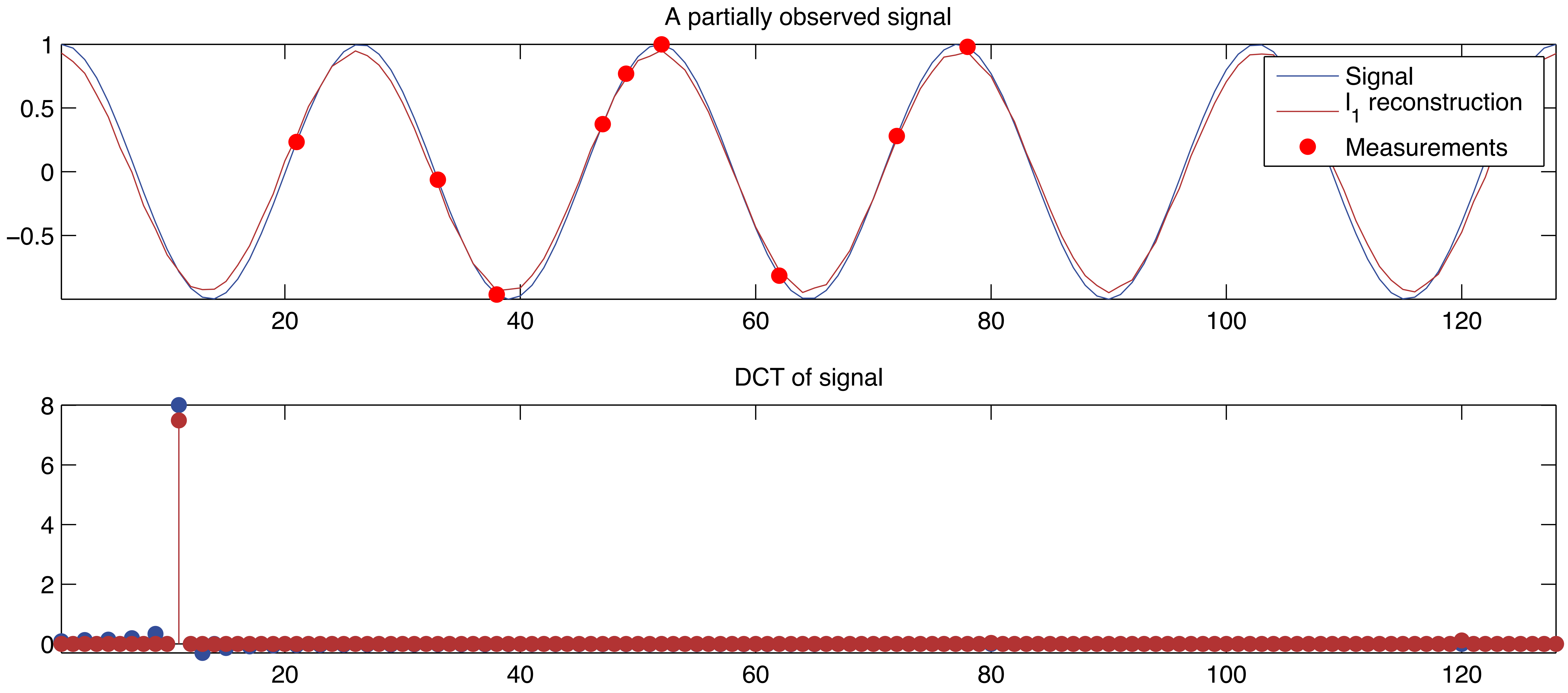


What happened?

- Minimizing the ℓ_2 resulted in adding more frequencies in the signal
  - ℓ_2 doesn't give sparsity
  - Small coefficients ≠ min ℓ_2
- We instead should find a minimal ℓ_1 solution
  - Because it actually enforces sparsity

And the result

- A much better reconstruction



A realization

- According to the rules of sampling this is impossible!

- What is the magic taking place here?
  - Why is sparsity special?

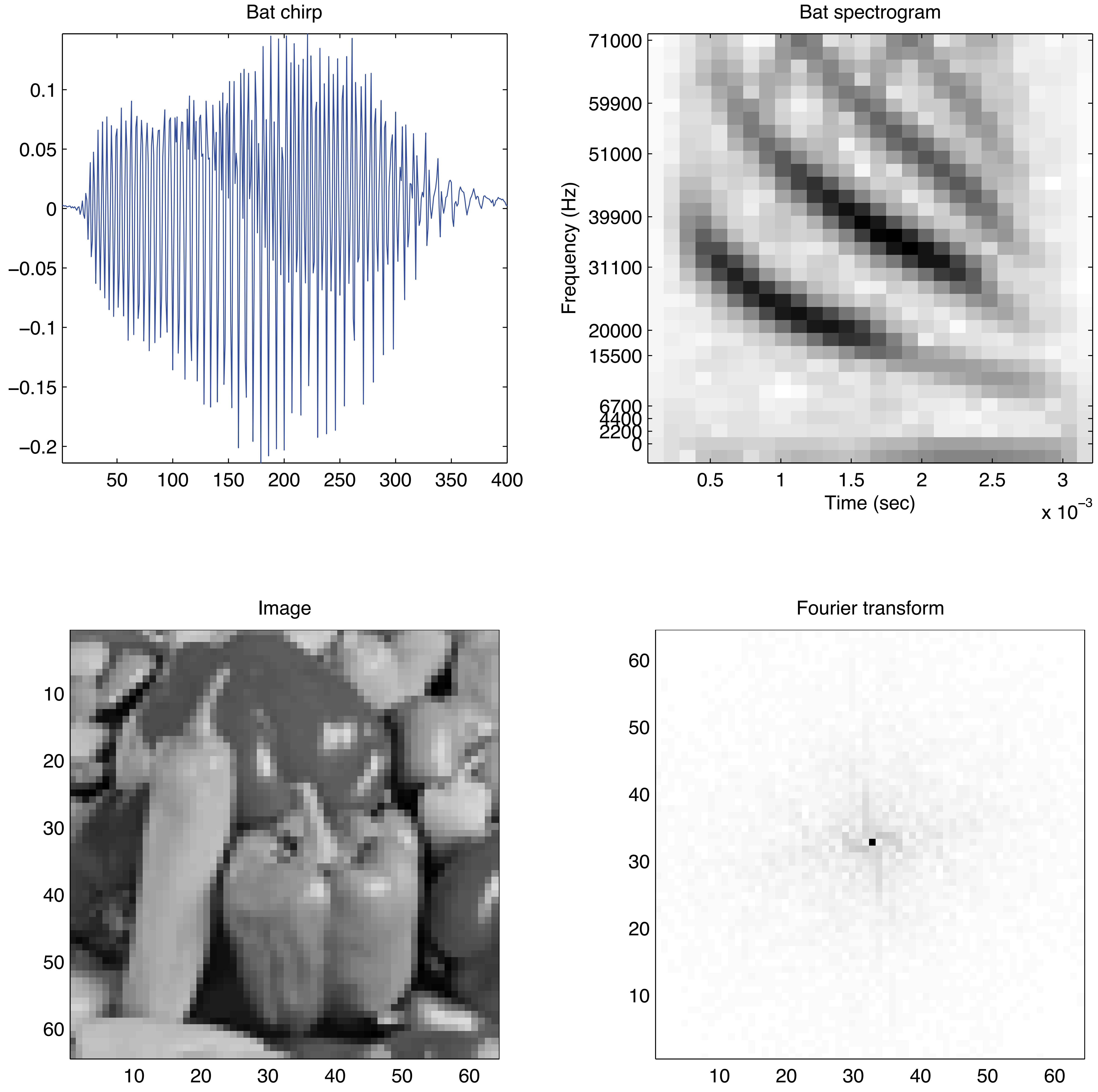
Why sparsity?

- Sparsity implies structure, and structure is everywhere

- Signals often exhibit sparsity after undergoing the right transformation

Sparsity in signals

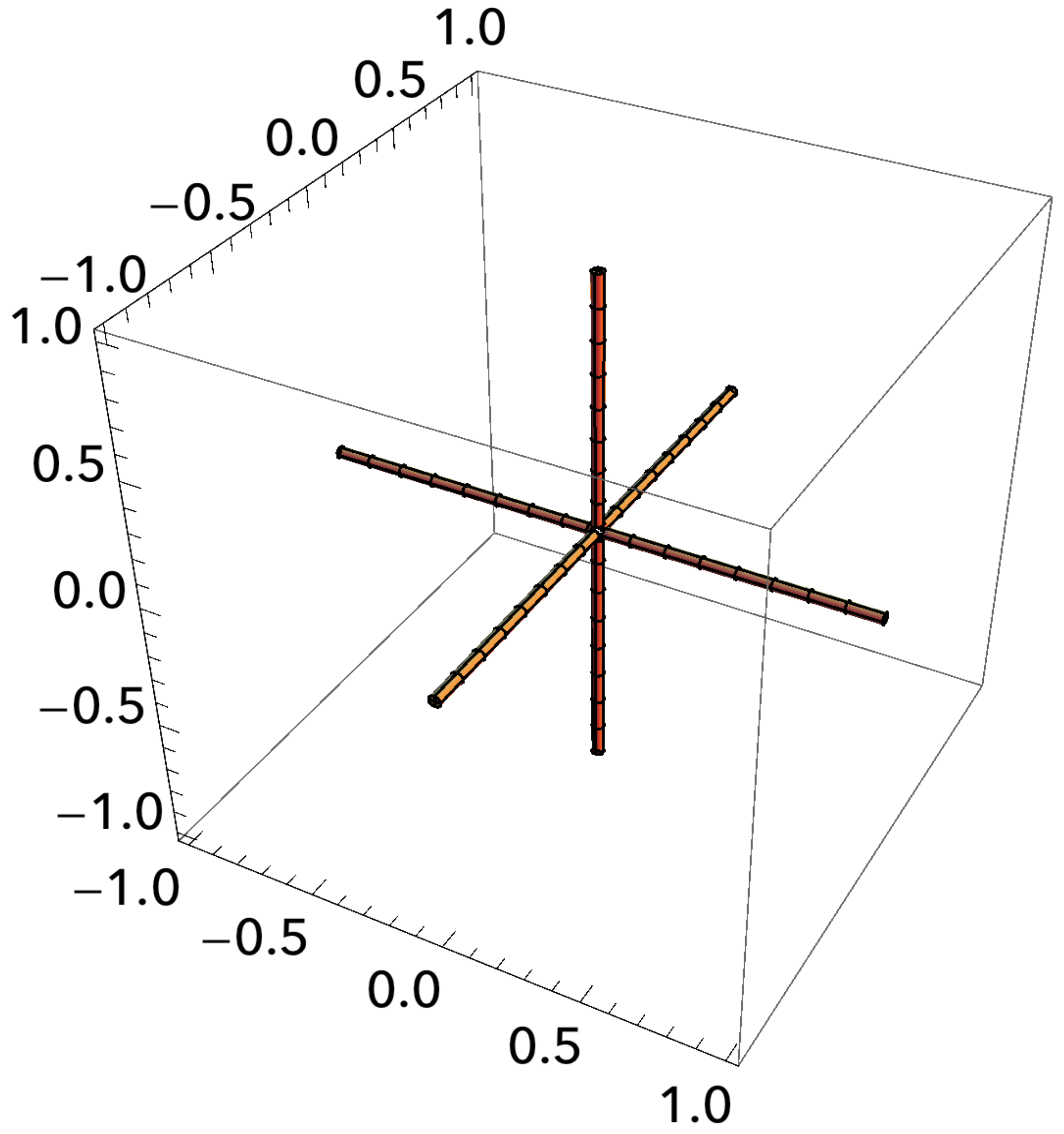
- Many signals are sparse in certain domains
  - e.g. sound spectra
  - Image wavelets
  - …
- This is an opportunity to measure signals better
  - or to describe them easier



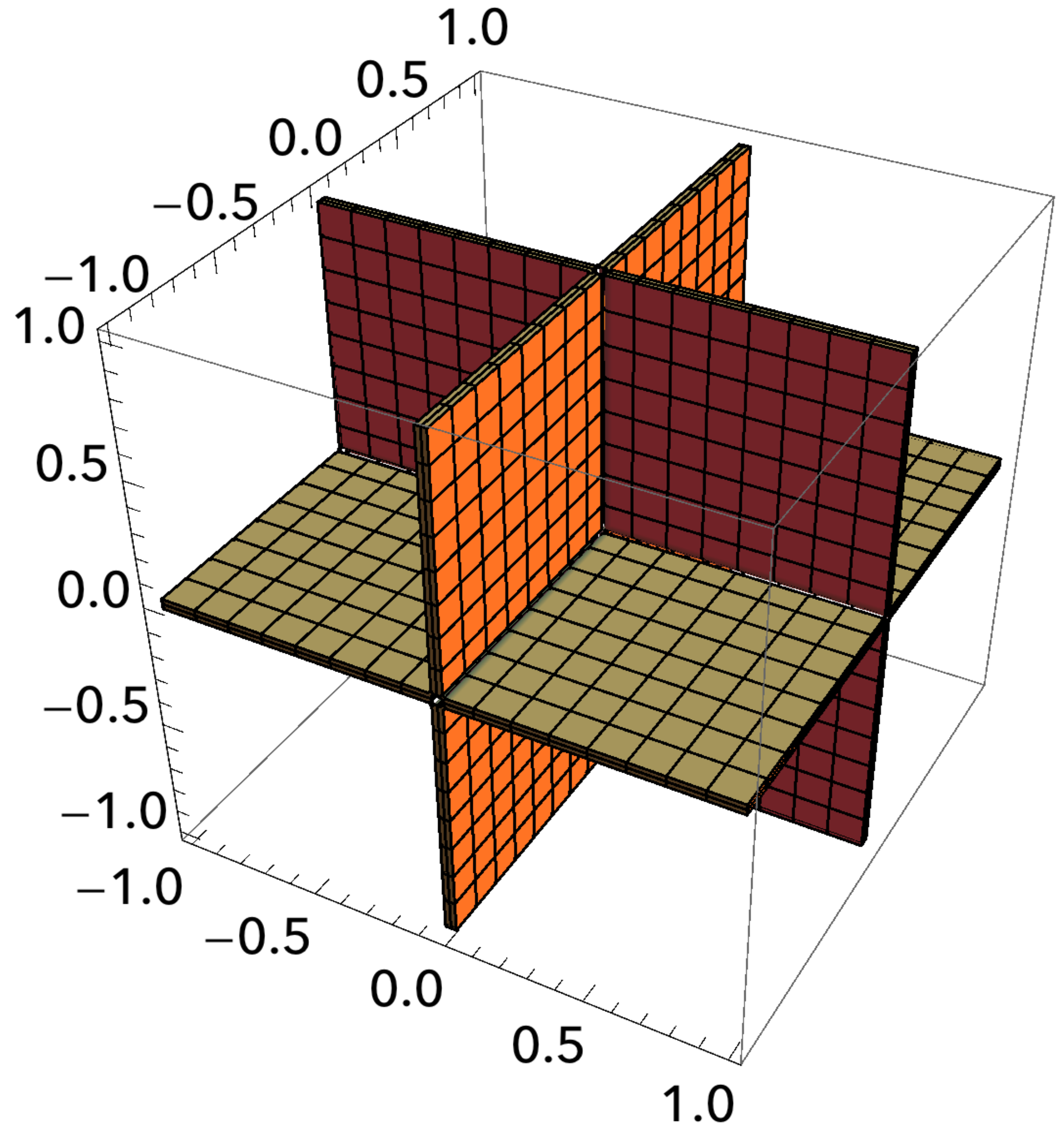
Vector spaces of signals

- Signals can be sparse in various ways
  - And some are "compressible"

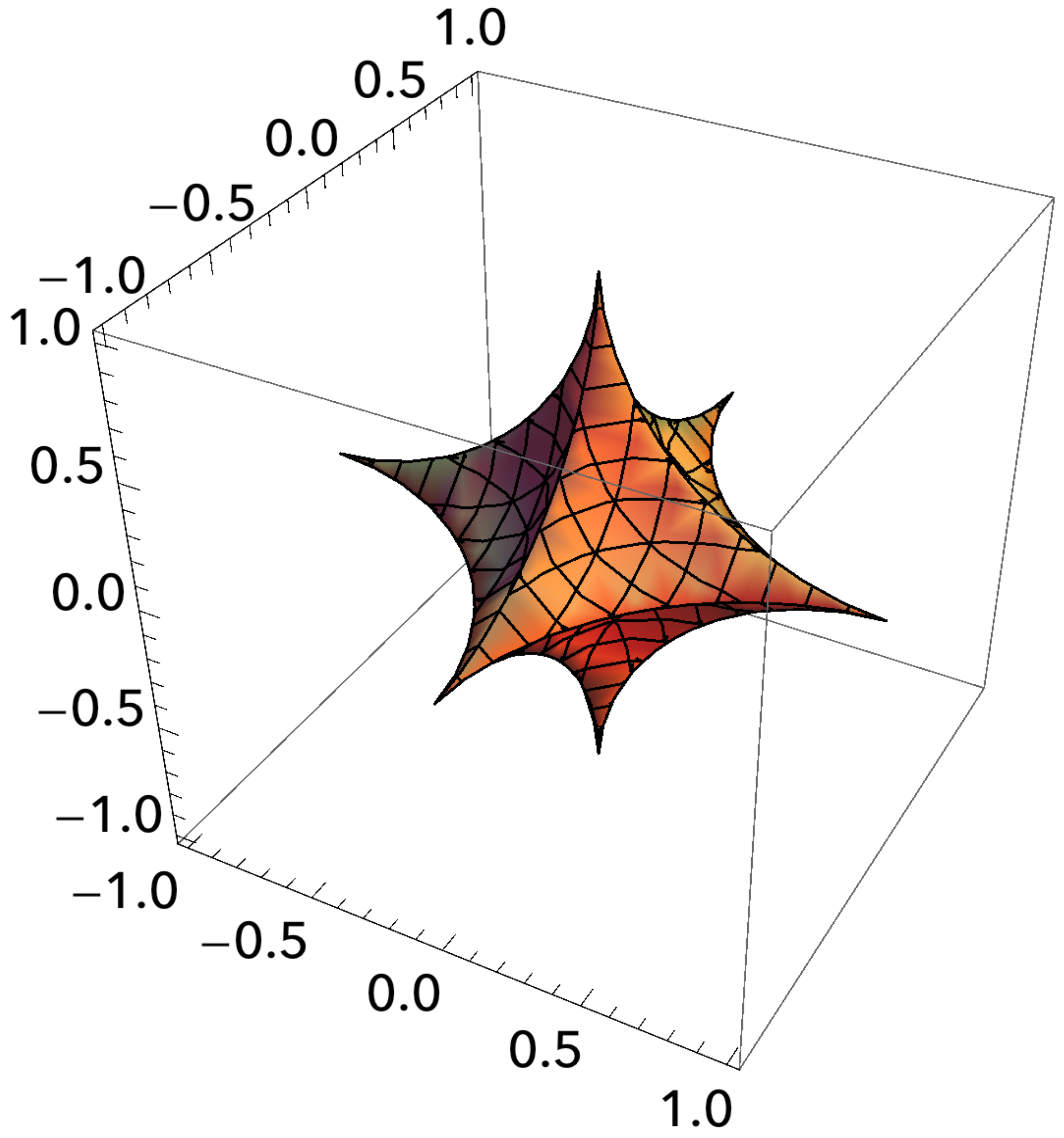
1-sparse signal space



2-sparse signal space



Compressible signal space



Sparse approximations

- Represent signals using:

Coefficients

$$\mathbf{f} = \sum_k a_k \mathbf{b}_k$$

Bases / Dictionary

- Two goals:
  - Analysis: Study $\mathbf{f}$ through structure of a and $\mathbf{b}$ (seen that)
  - Approximation: Reconstruct $\mathbf{f}$ with a minimal number of terms

Exposing sparsity via dictionaries

- Can we use dictionaries that produce sparse coefficients?

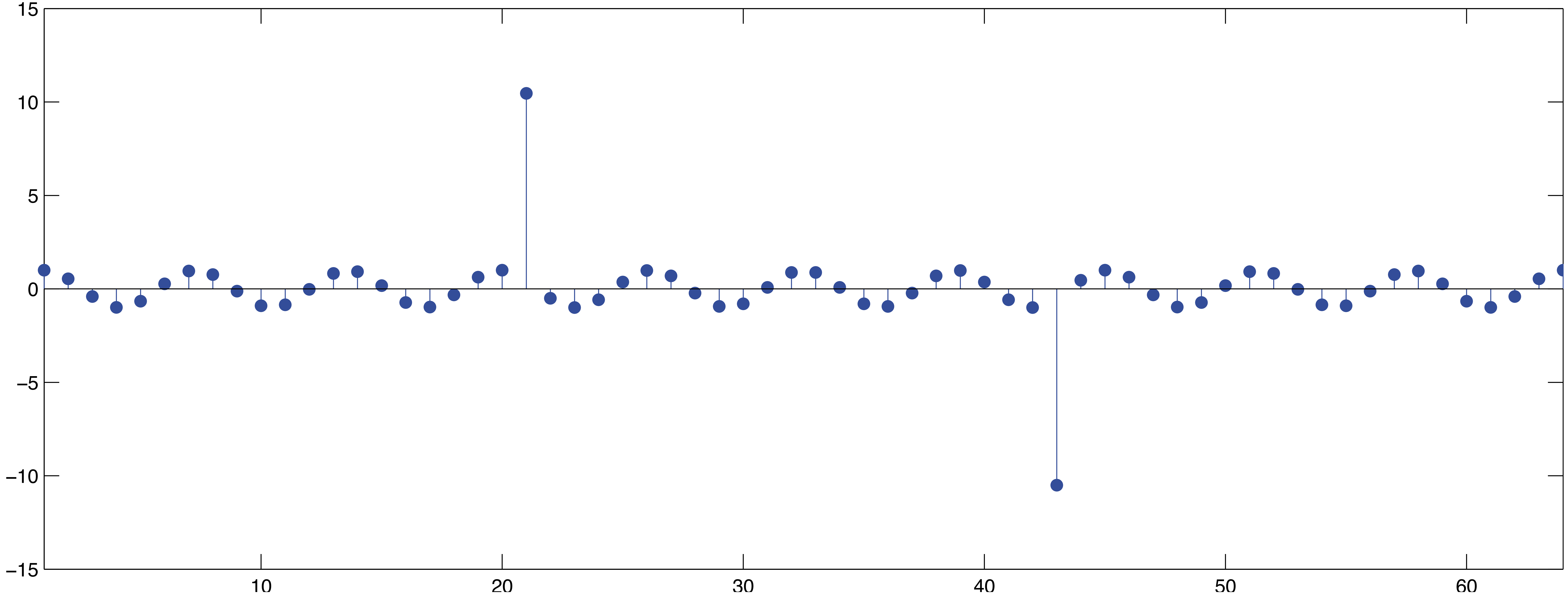
- Why would they be useful and how would we implement them?

Example case: Wavelets

- Note how most coefficients are zero
  - A sparse representation

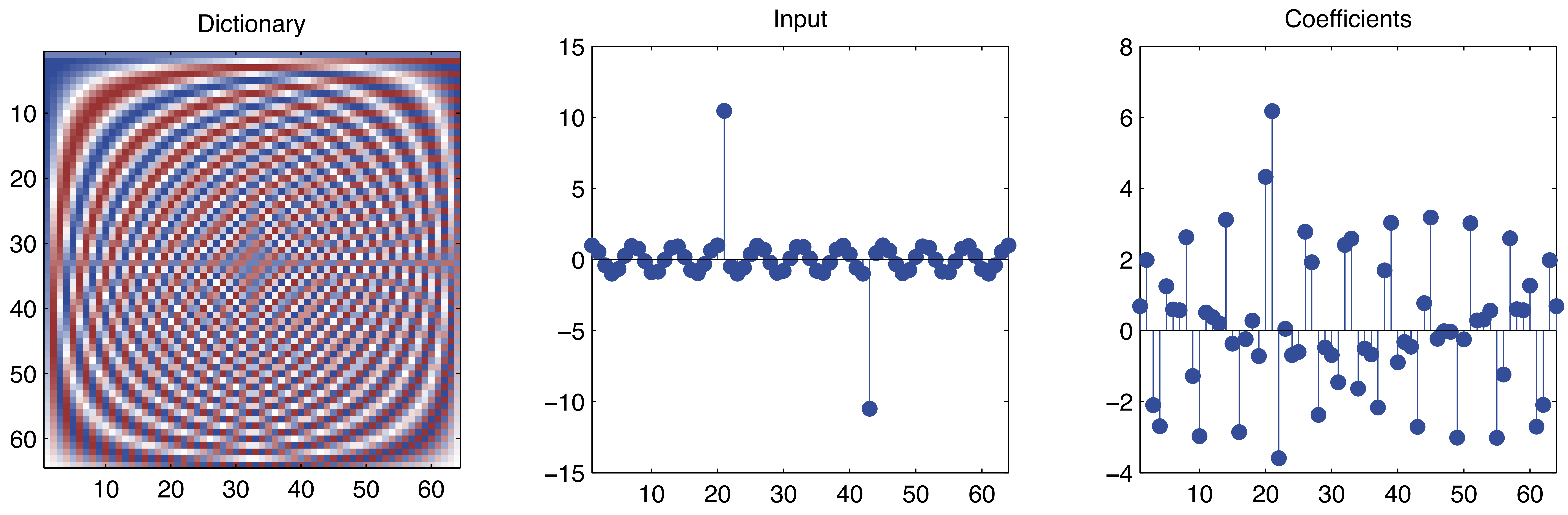
A simple example

- A sinusoid with a couple of spikes



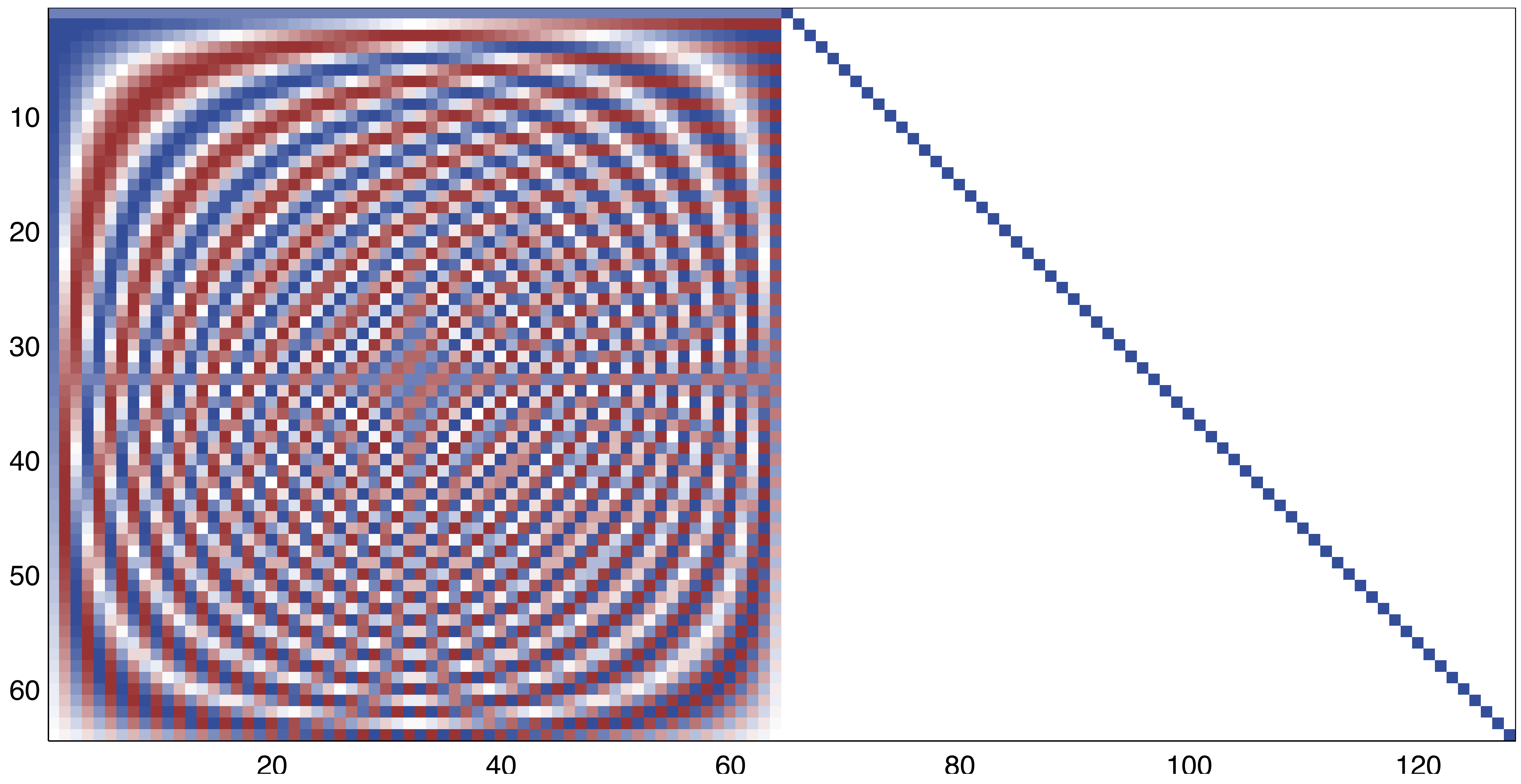
Using a generic dictionary

- Analyzed via the DCT
  - Resulting coefficients are not sparse
    - Multiple sines are used to approximate the spikes



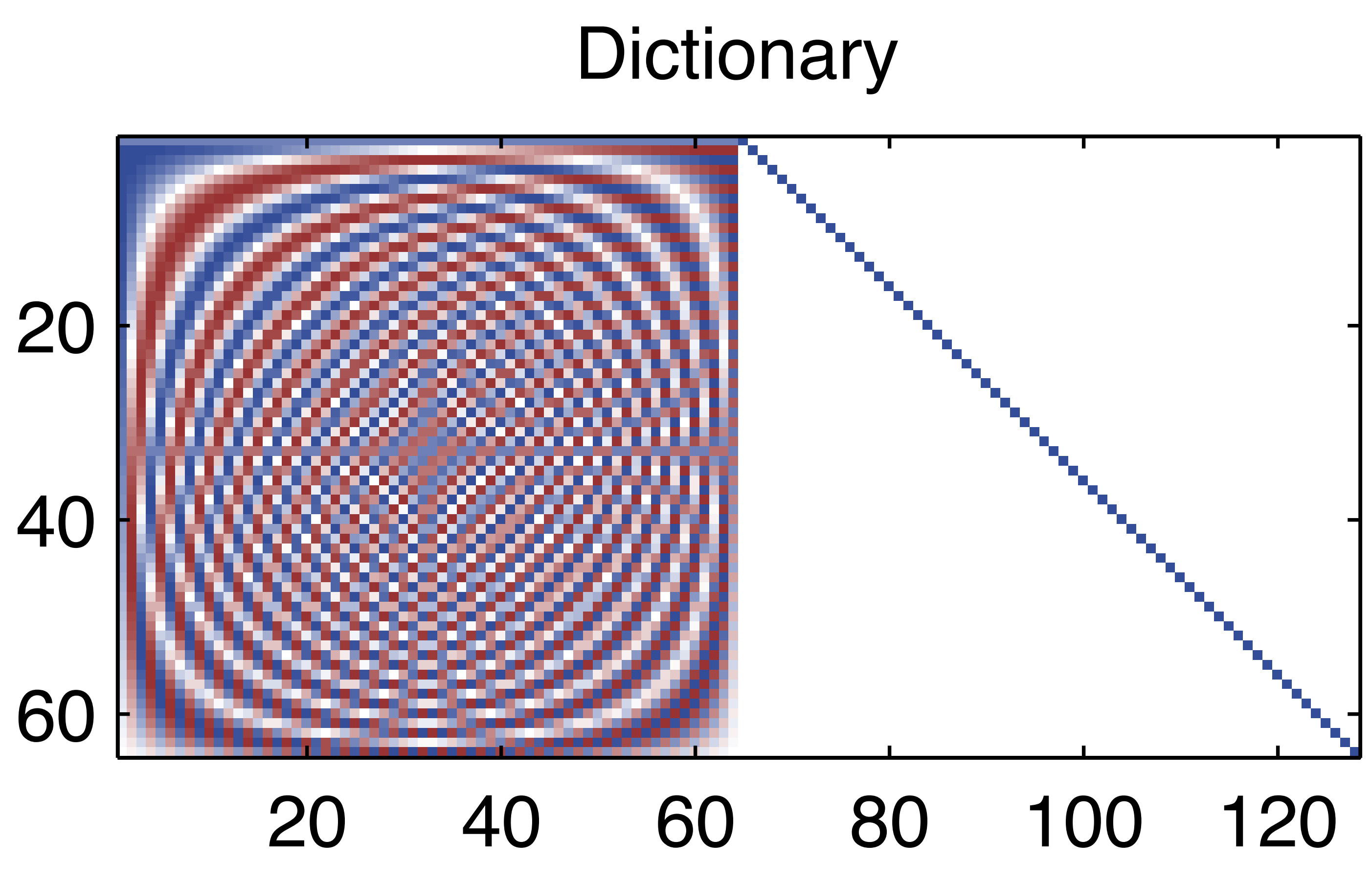
A “better” dictionary

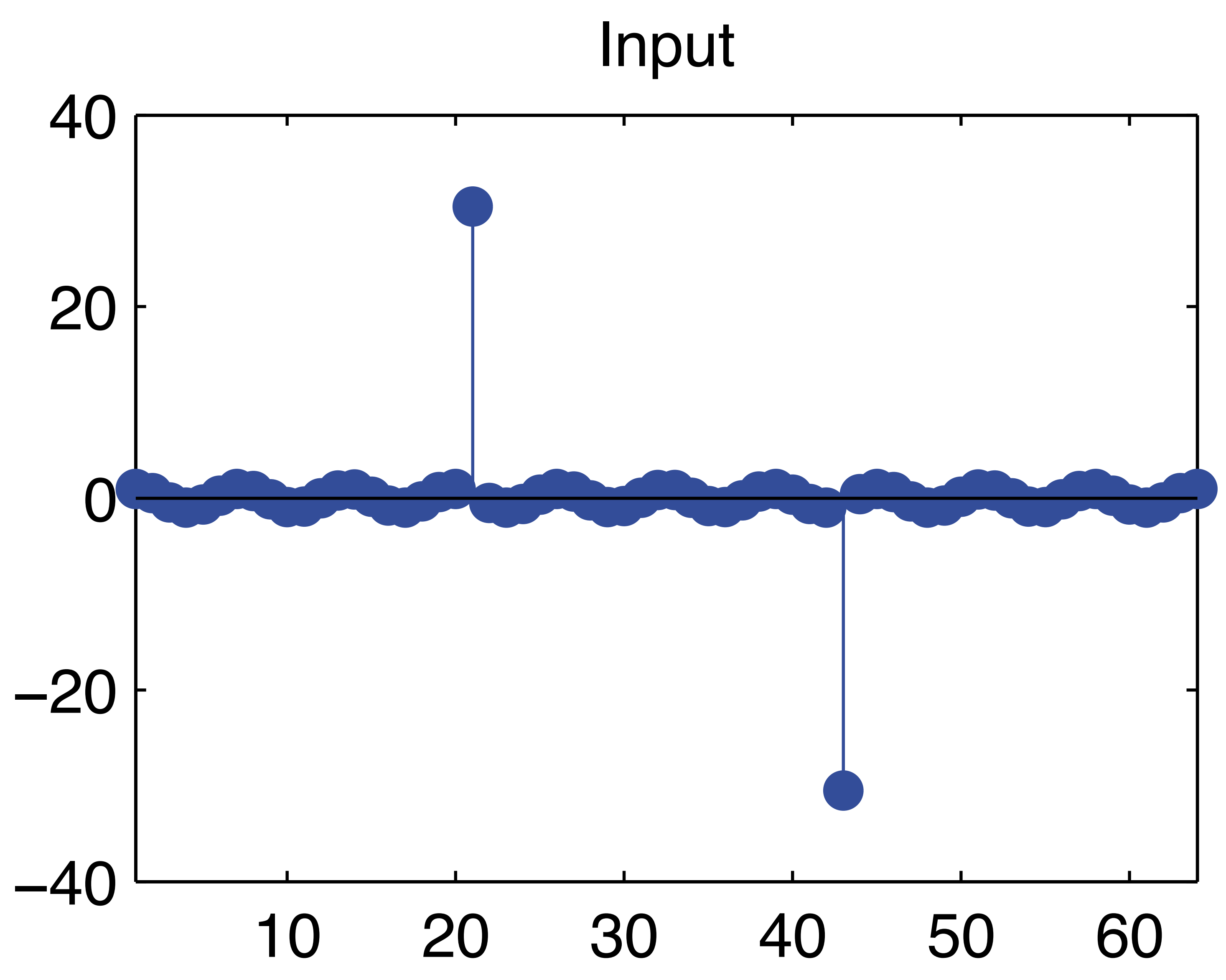
- Use both sinusoids and spikes!
  - Now we won’t use as many sines to represent the spikes

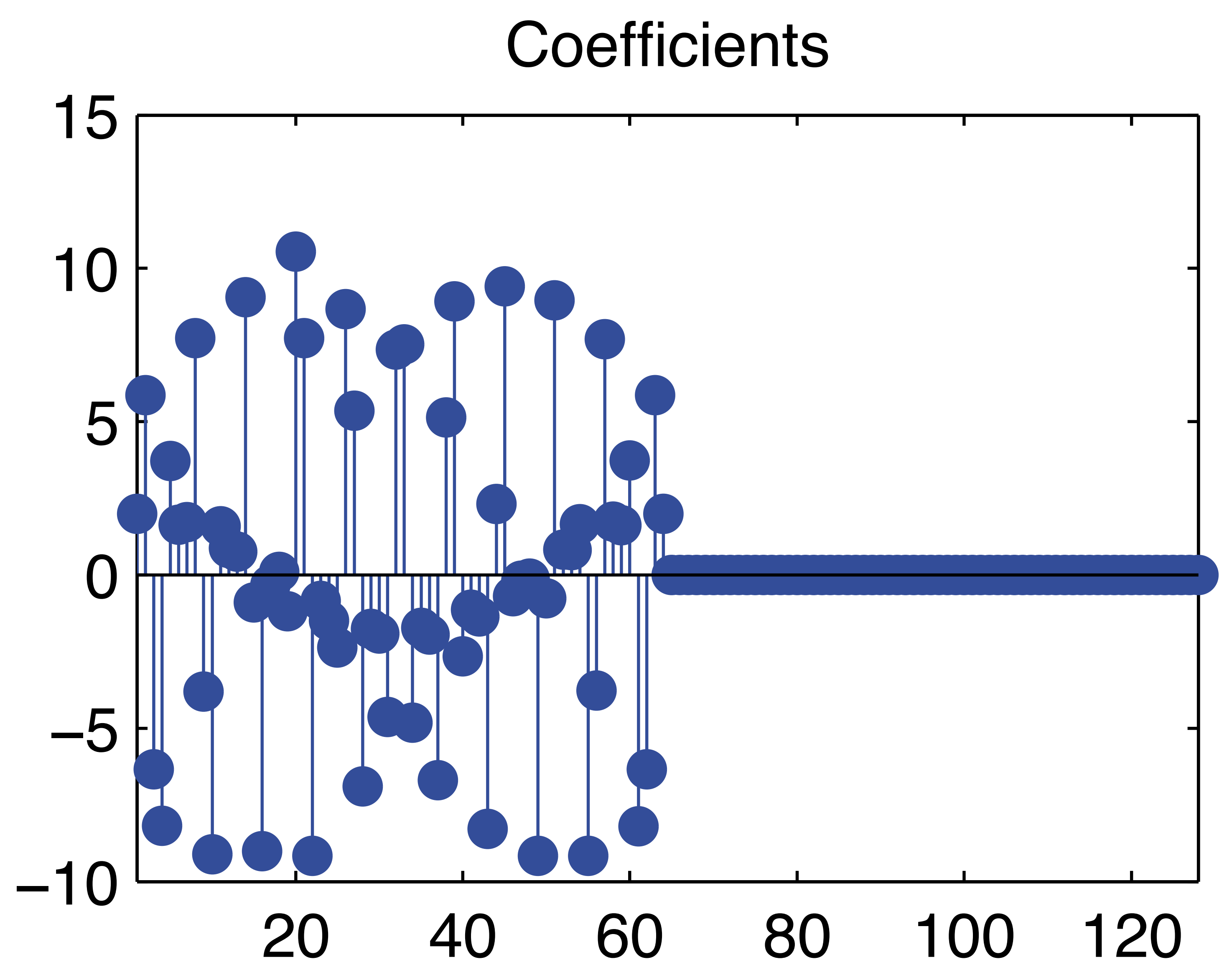


Applying the dictionary

- Using the straightforward decomposition won't help
  - Spike elements are not utilized
  - Minimal ℓ_2 cost penalizes the bases describing the loud spikes

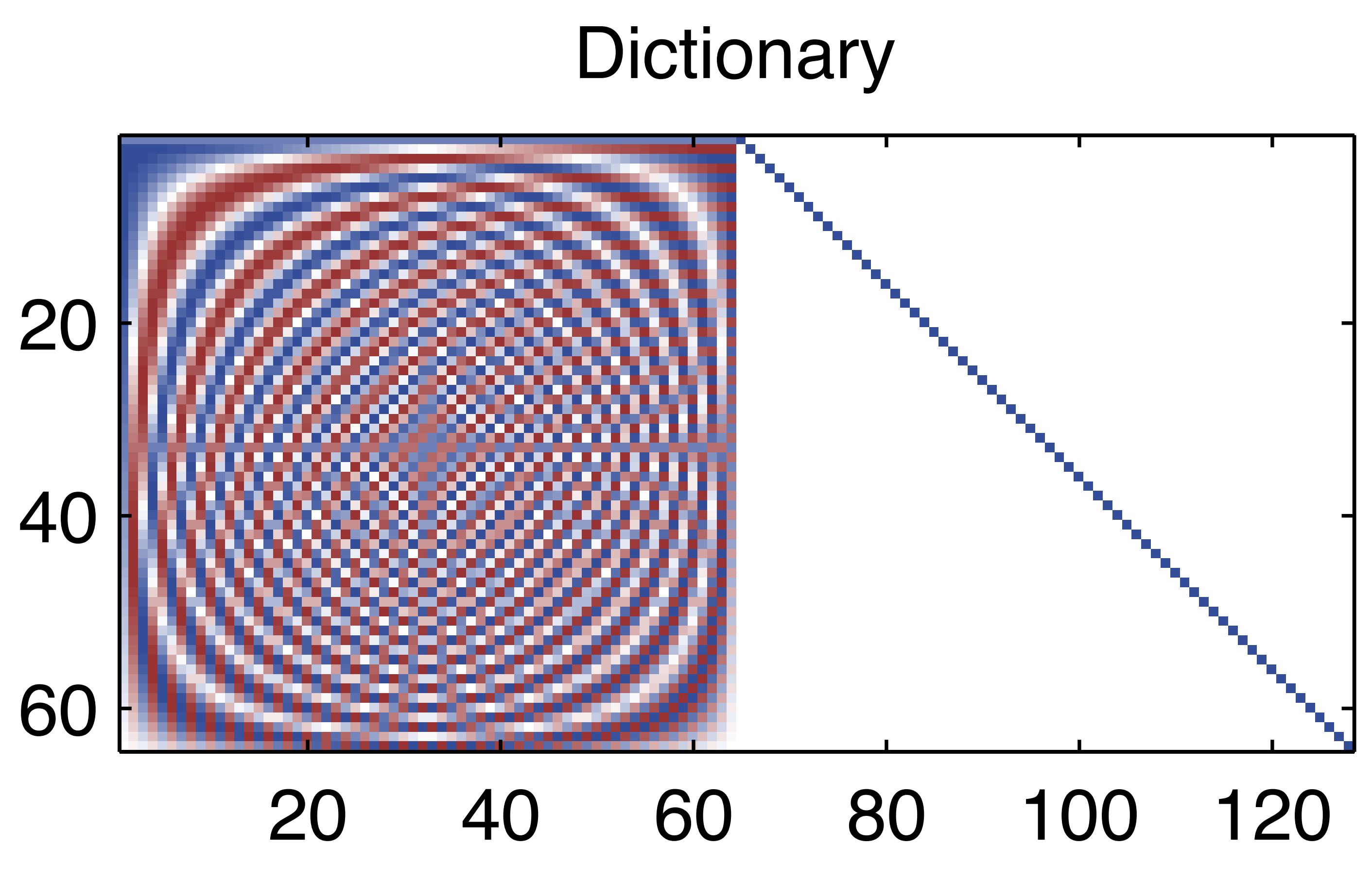


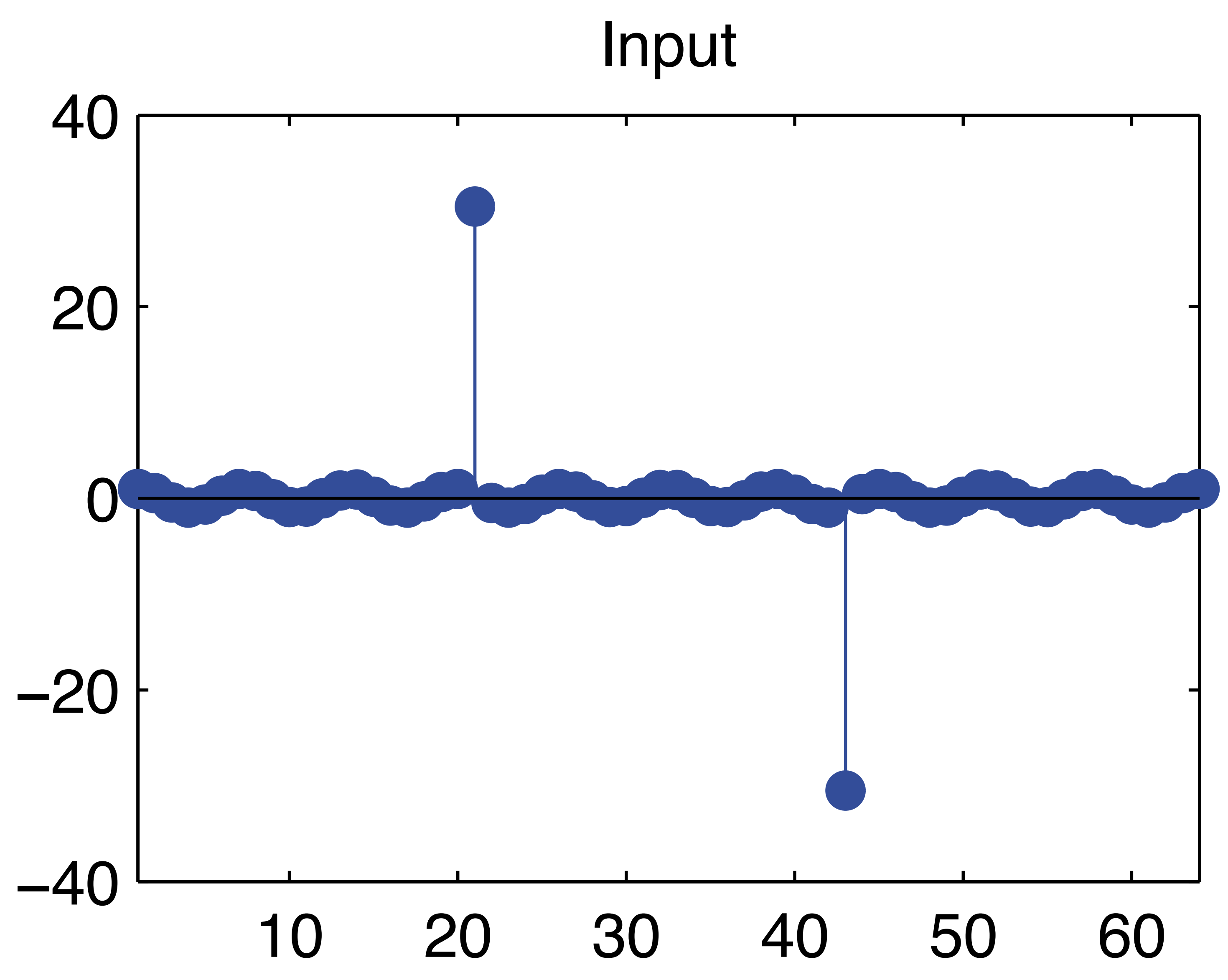


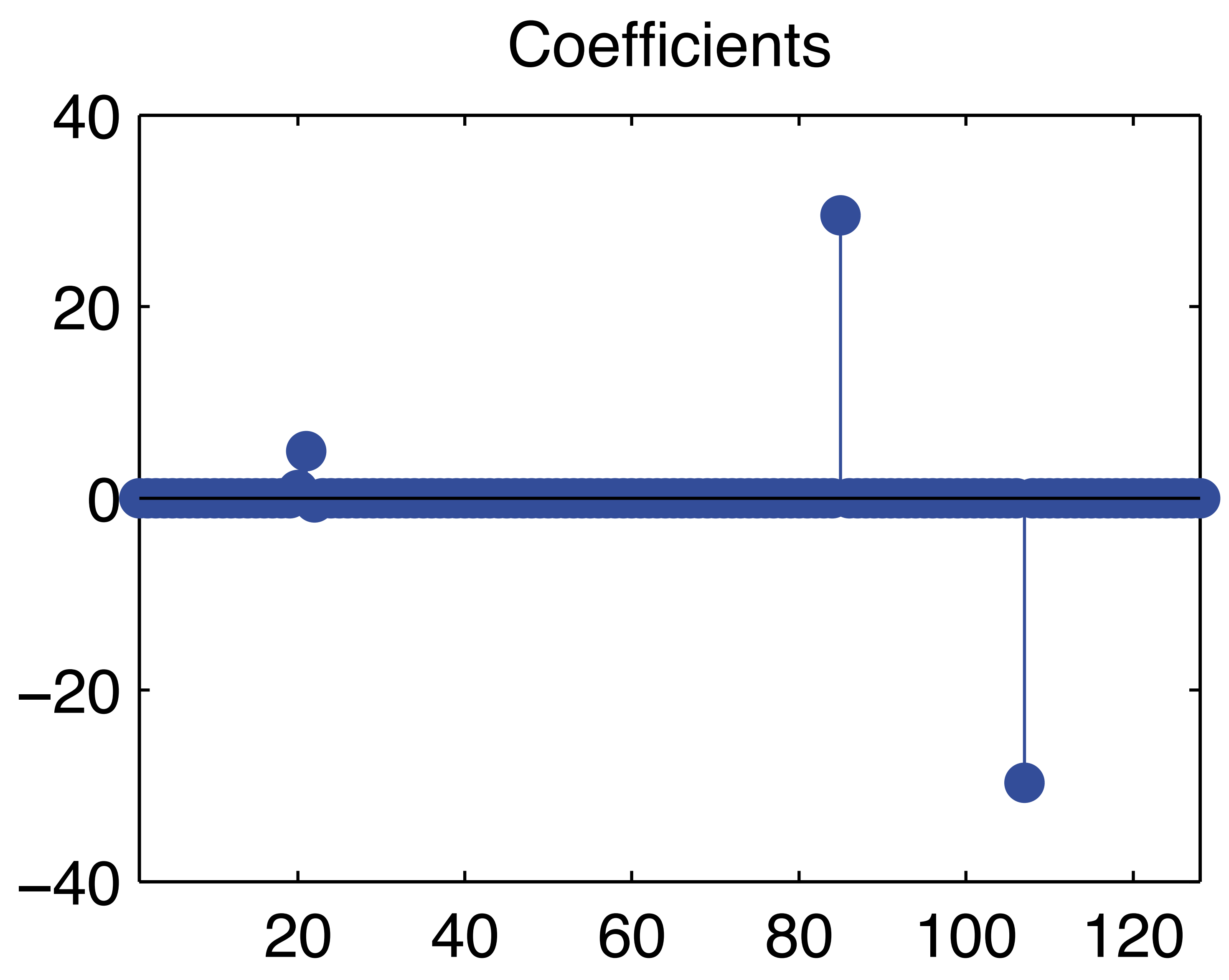


Doing it the right way

- This time we ask for minimum ℓ_1 coefficients
  - And we get a perfect description of the input!







Overcomplete dictionaries

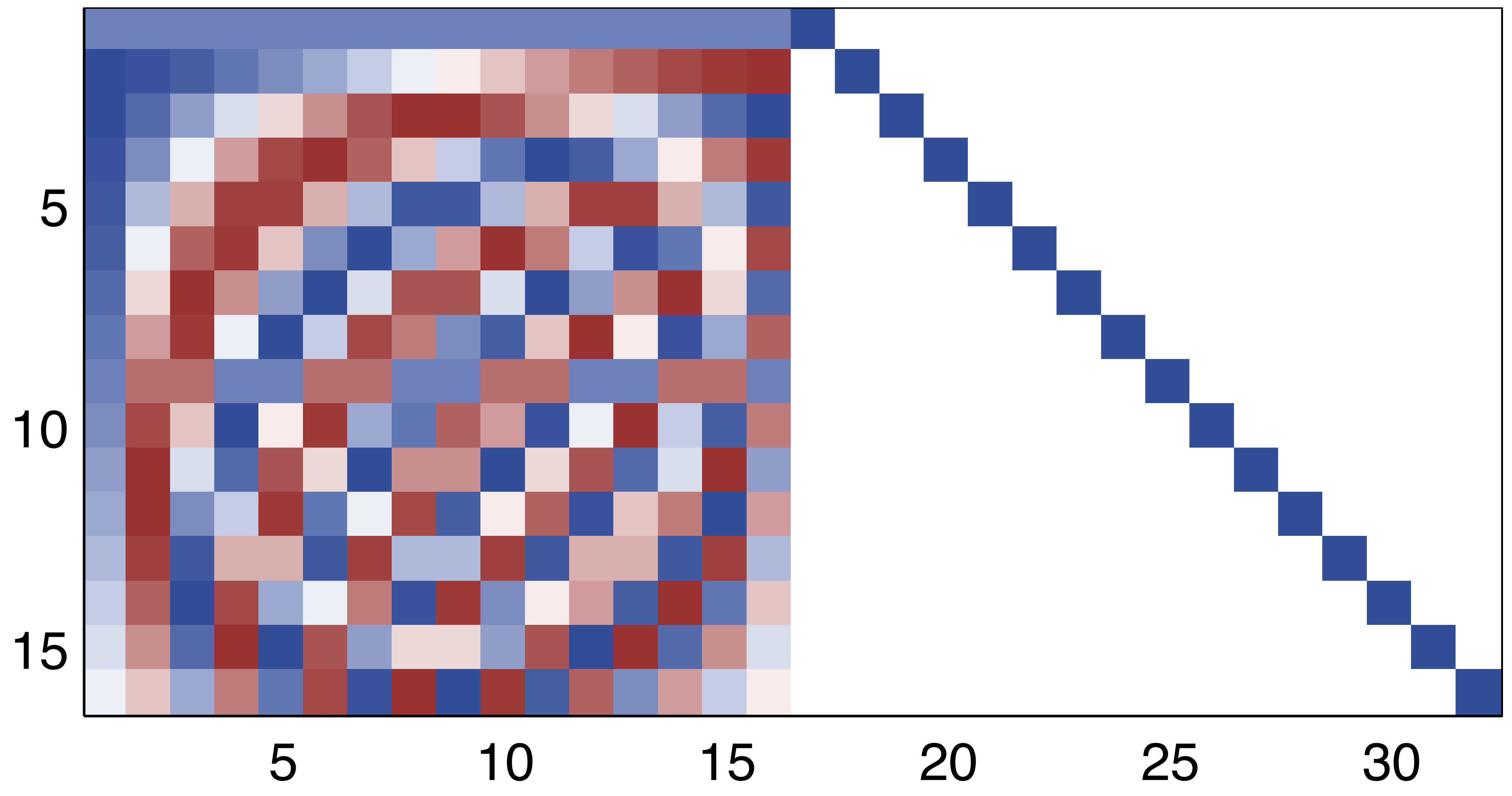
- Use dictionaries that contain “everything”!
  - Use compact descriptions of elements

- Some problems
  - Large size/computations
  - Lack of fast algorithms (e.g. FFT)
  - Problems with “coherence”
    - This is a big one

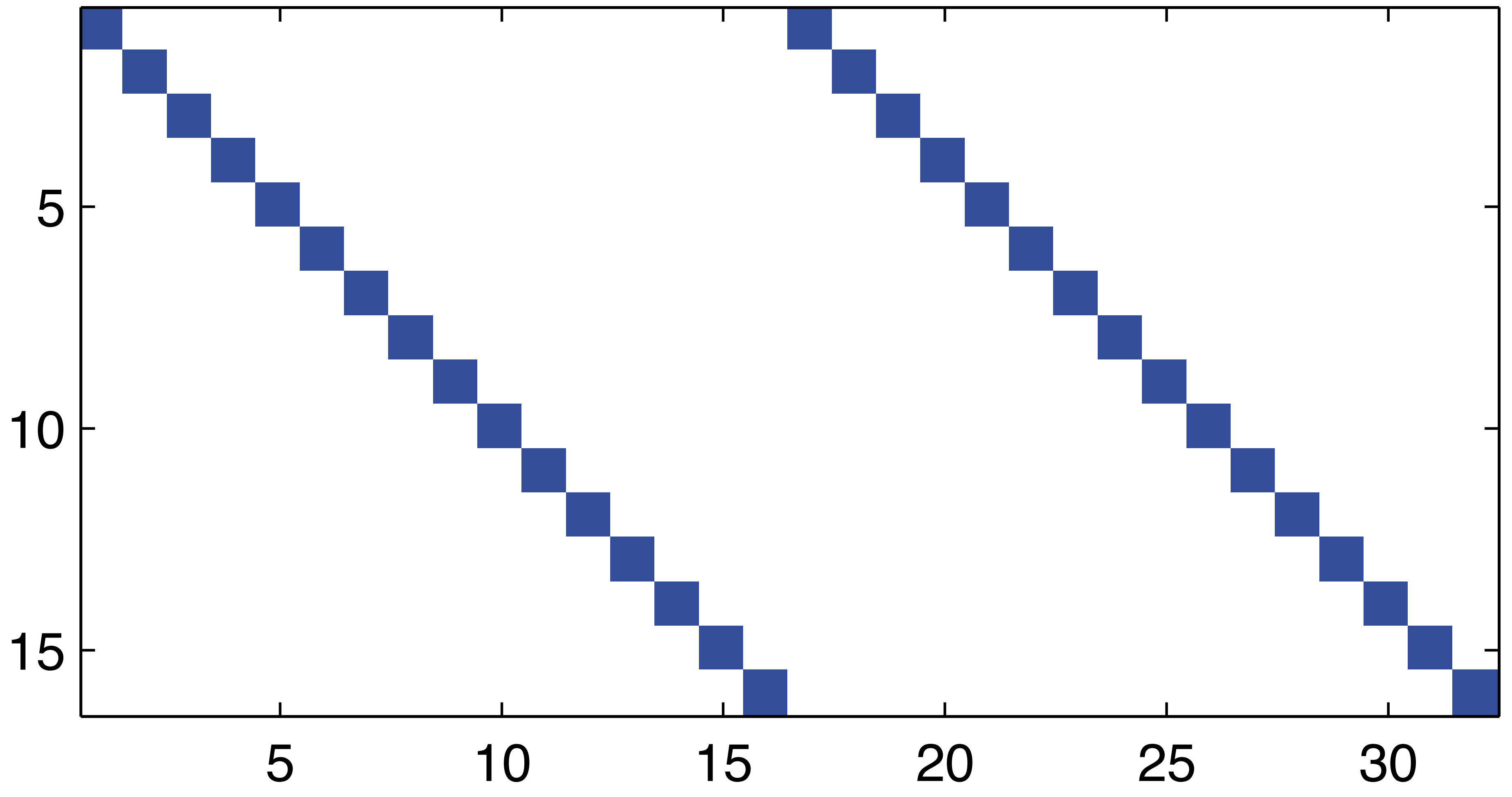
Dictionary coherence

- Make sure that the dictionary elements don't result in ambiguous coefficients
  - A measure of coherence: $\mu = \max_{i,j} \left| \left\langle \mathbf{d}_i, \mathbf{d}_j \right\rangle \right|$

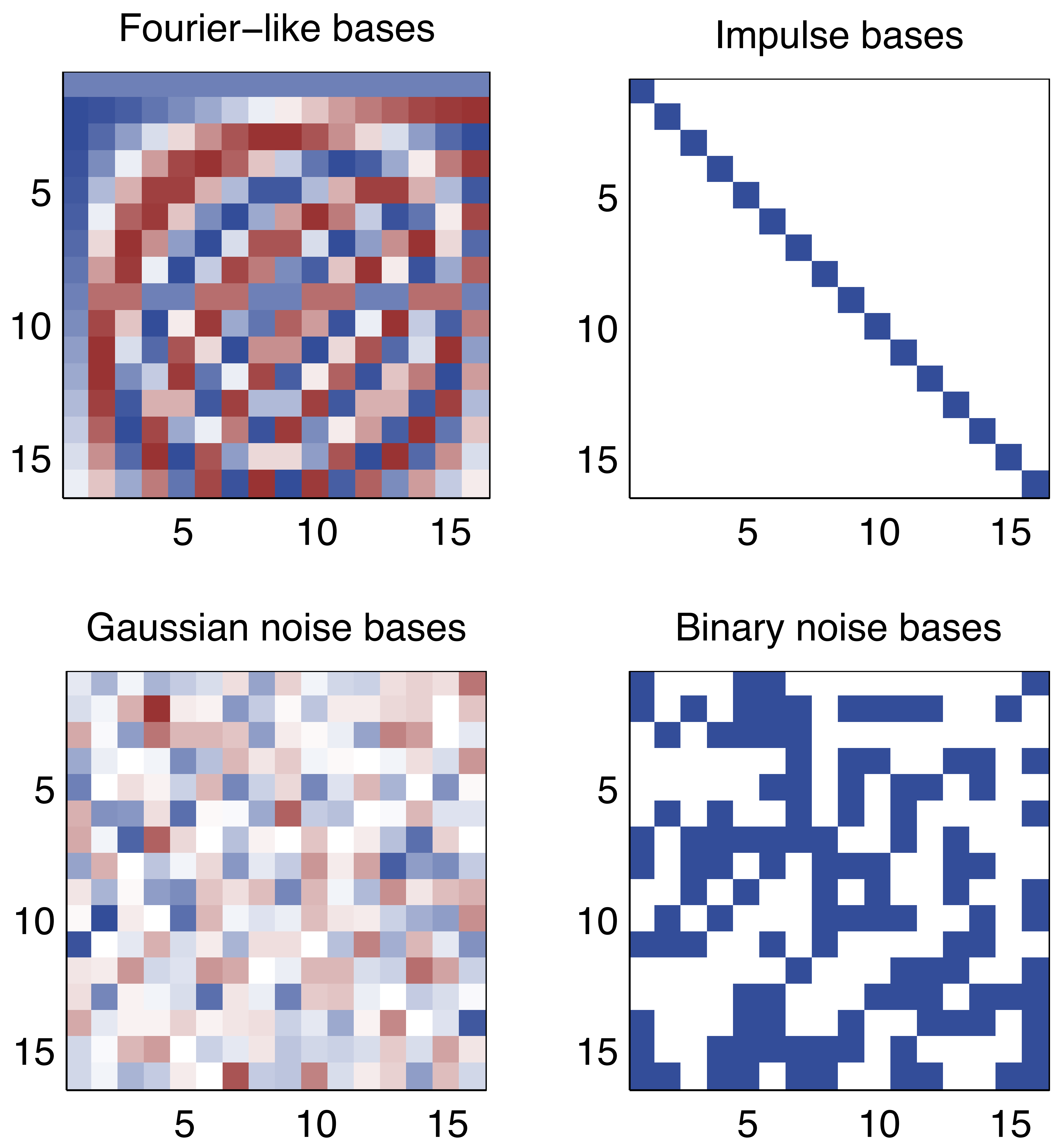
Good



Bad!



Examples of incoherent dictionaries

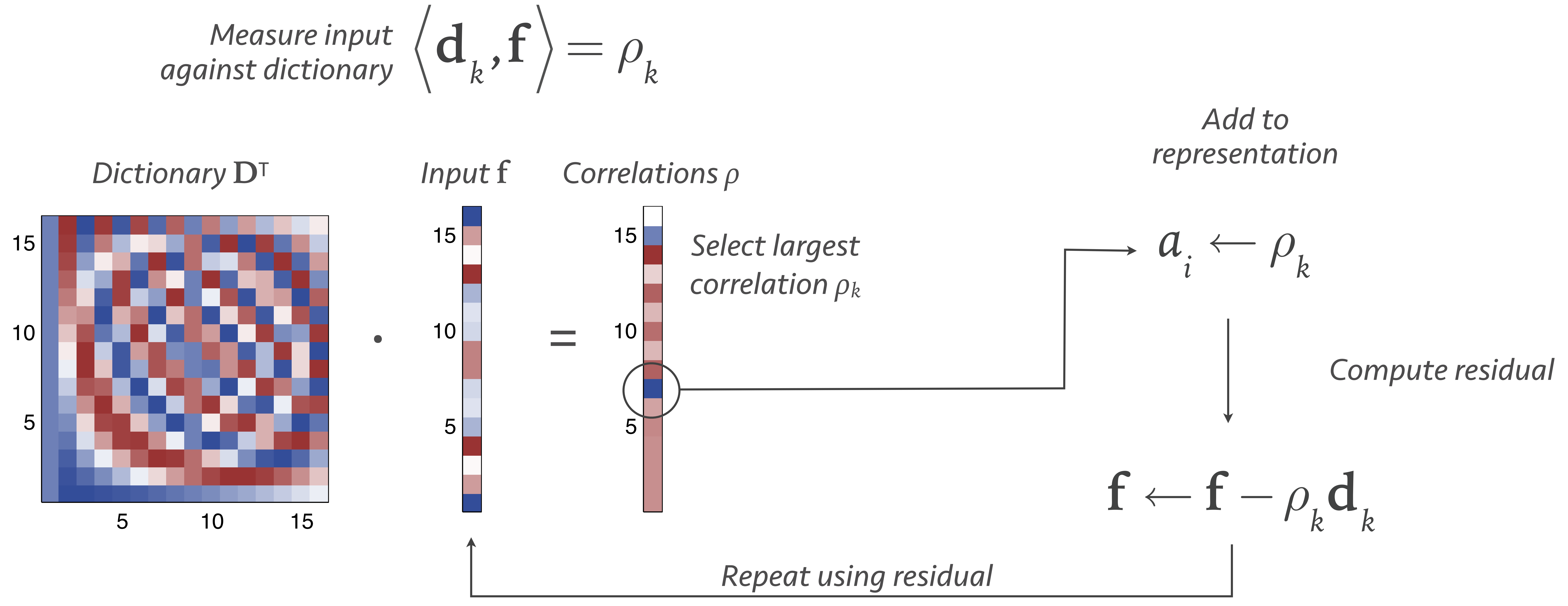


Getting greedy

- The linear programming approach will fail now
  - It is slow for large dictionaries
  - It looks for exact equality, not approximation
- We can instead use a greedy approach to resolving sparse approximations

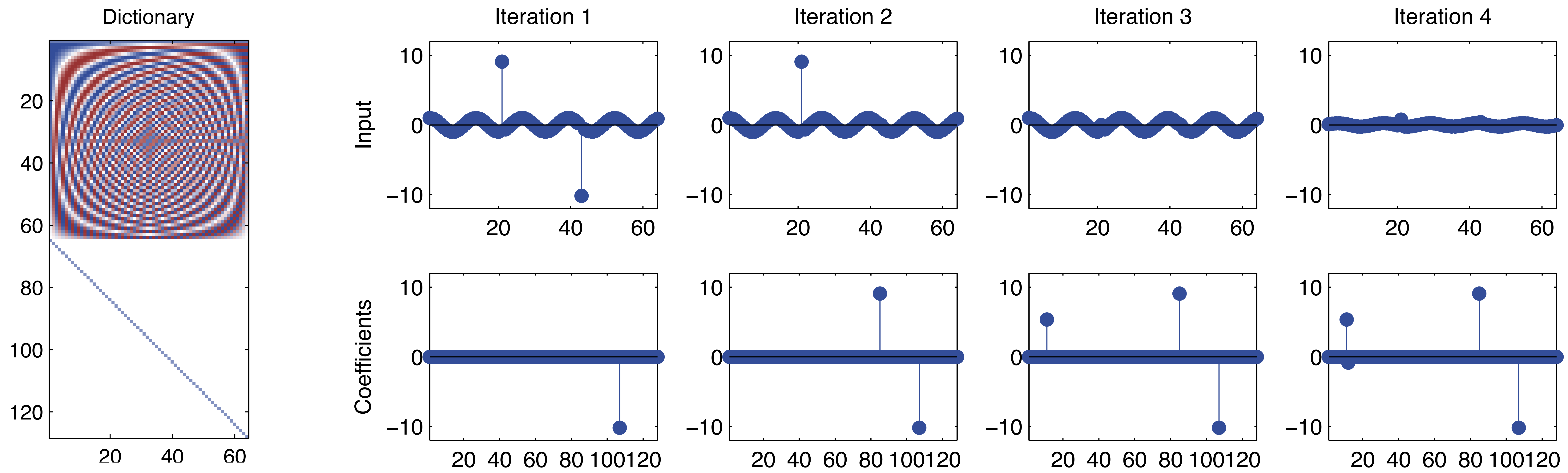
Matching pursuit (MP)

- Family of many approaches based on successive fits
  - Each new fit explains what's the previous ones couldn't



On a familiar example

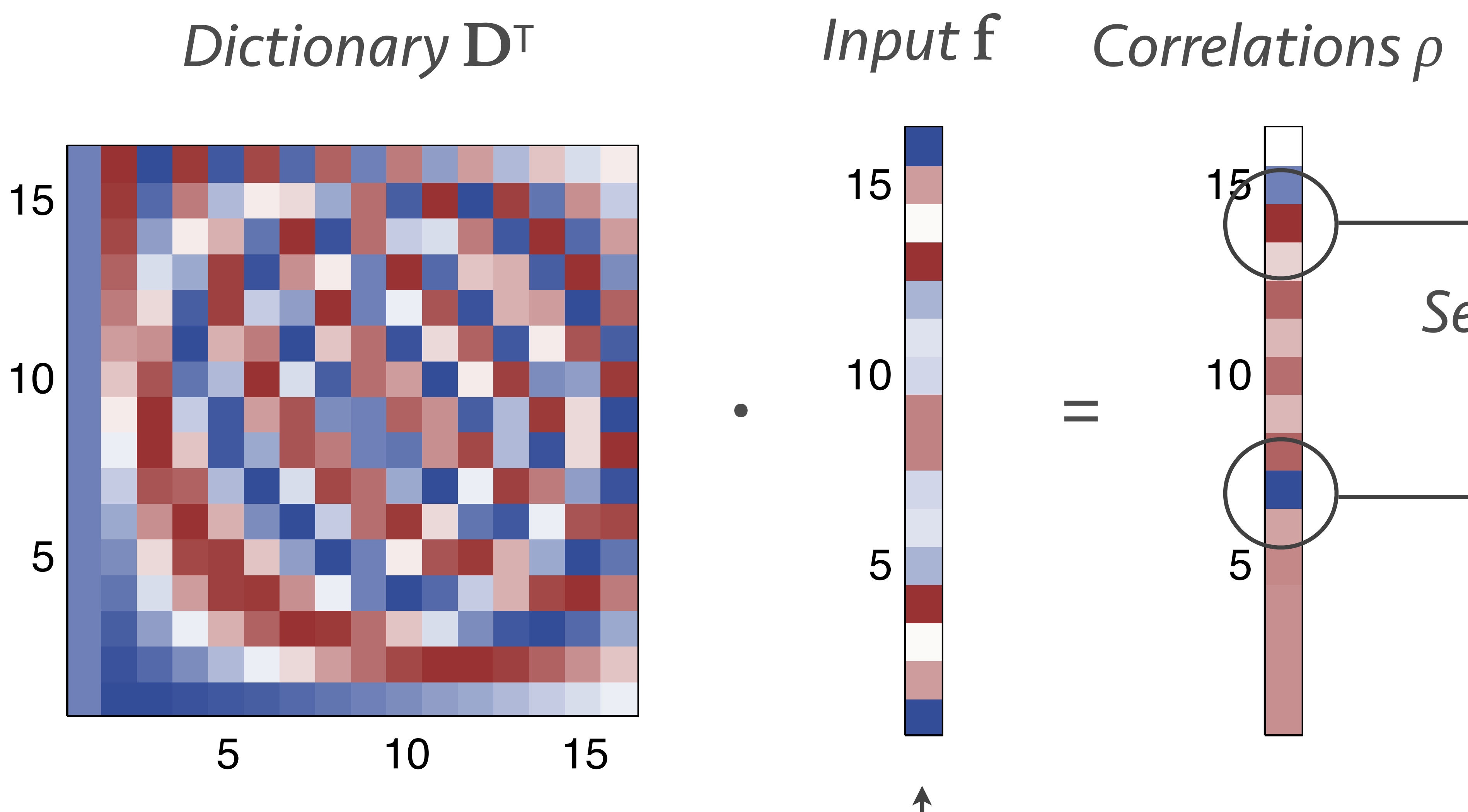
- Each iteration knocks off an element
  - by 4th iteration there's nothing significant left to represent
    - Faster! For $N = 1024$, MP: 0.005 sec, LP: 63 sec



CoSaMP (Compressive Sensing MP)

- A useful variation to get K non-zero coefficients

Measure input against dictionary $\left\langle \mathbf{d}_k, \mathbf{f} \right\rangle = \rho_k$



Select locations of largest $2K$ ρ_k

Add to support set

$$\Omega = \mathrm{supp}\left(\rho\,|_{2K}\right) \cup T$$

Invert over support

$$\mathbf{b} = \mathbf{D}_{\Omega}^{\dagger} \cdot \mathbf{f}$$

Truncate to K and compute residual

$$T = \mathrm{supp}\left(b\,|_K\right)$$

$$a = b\,|_K$$

$$\mathbf{f} \leftarrow \mathbf{f} - \mathbf{D} \cdot a$$

Repeat using residual

Revisiting sampling

- Traditional acquisition samples uniformly
  - e.g. constant sample rates in audio, CCD grids in camera
- Foundation: Nyquist/Shannon sampling theory
  - Sample at twice the highest frequency
  - Projects to a complete basis that spans all the signal space

A redundancy in the loop

- Take a picture
  - Using dense sampling → lots of data
- Transform to a sparse domain and quantize
  - i.e. MPEG/JPEG compression → fewer data
- Process, transmit, view, etc.

Compressive sensing

- Why sample and then compress?
  - Do both at once!
- Sample fewer samples and use signal sparsity
  - Helps in finding a unique and plausible sparse signal

The compressive sensing pipeline

- Acquire signal using underdetermined measurements
  - e.g. linear combinations of a few samples: $\mathbf{y}_i = \mathbf{P}_i \cdot \mathbf{x}$
    - Don't sample densely, don't sample all the data
- Reconstruct signal assuming sparsity
  - Sparsity constraints signal space w.r.t. measurements
  - Allows for a plausible reconstruction

What's a good measurement matrix?

- We need:

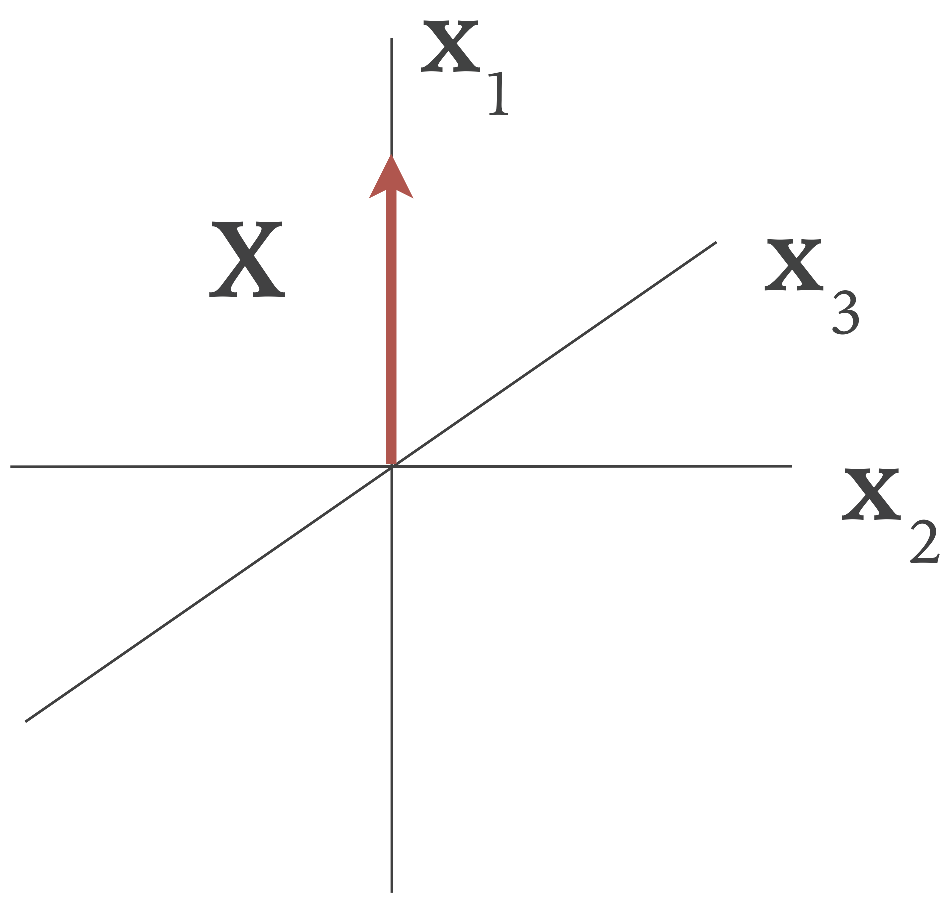
$$\mathbf{P} \cdot \mathbf{x}_1 \neq \mathbf{P} \cdot \mathbf{x}_2 \quad \text{for all } K\text{-sparse } \mathbf{x}_1 \neq \mathbf{x}_2$$

  - i.e. ensure that we can distinguish different inputs

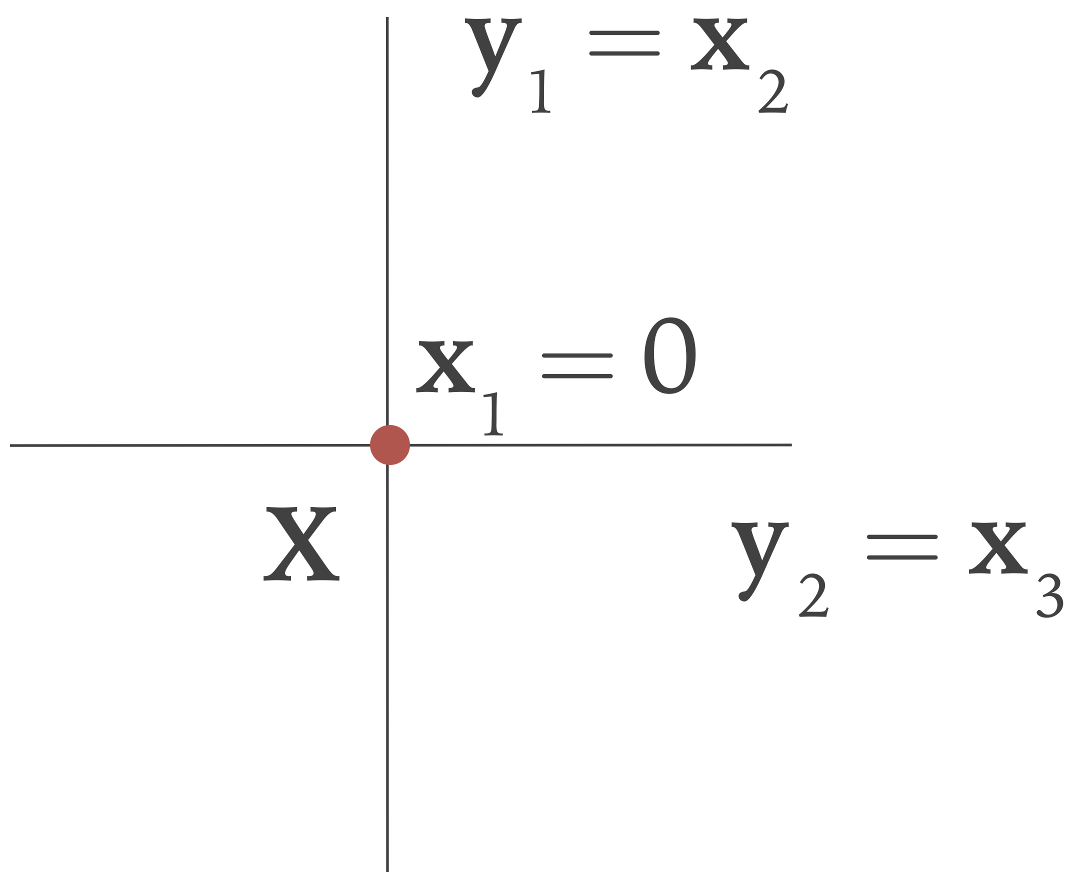
- Necessary condition: $\mathbf{P}$ must have at least $2K$ rows
  - Assuming noiseless and well-behaved data

Example 1-sparse case

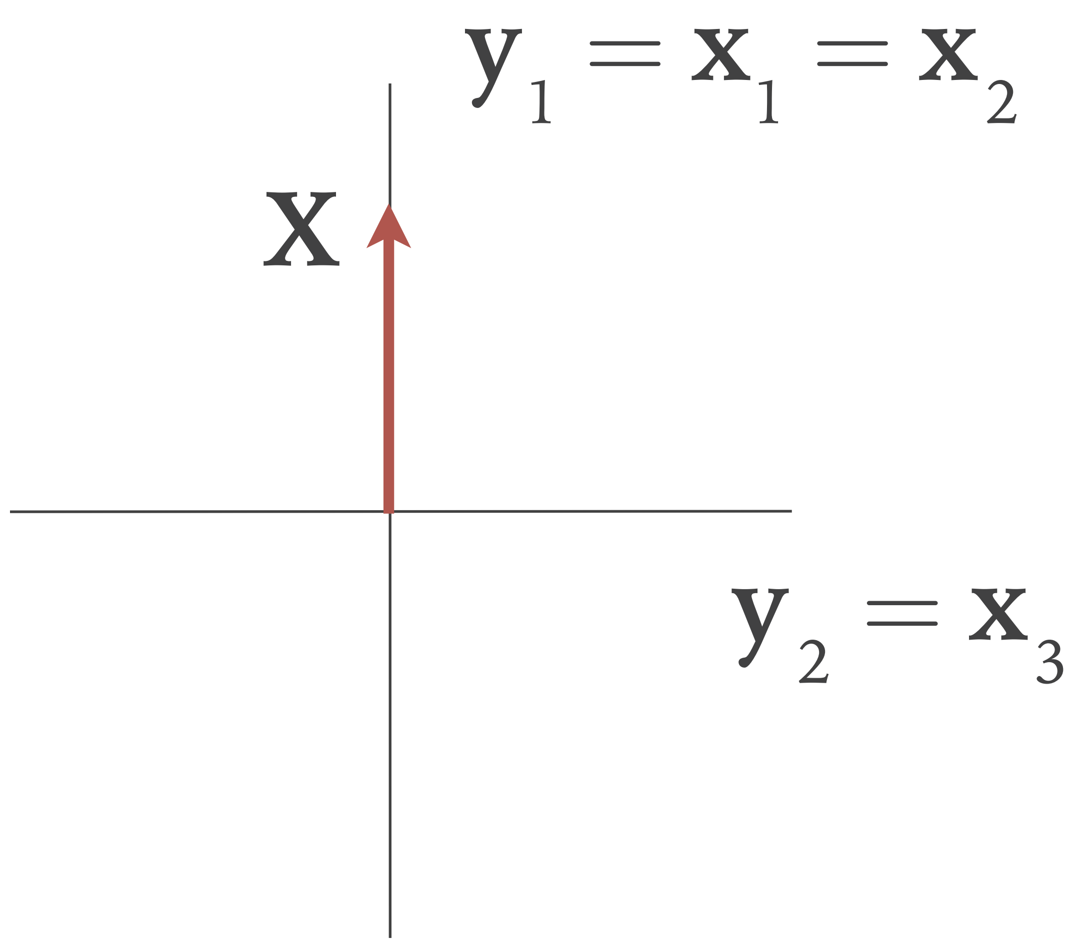
- Some poor measurement matrices



$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

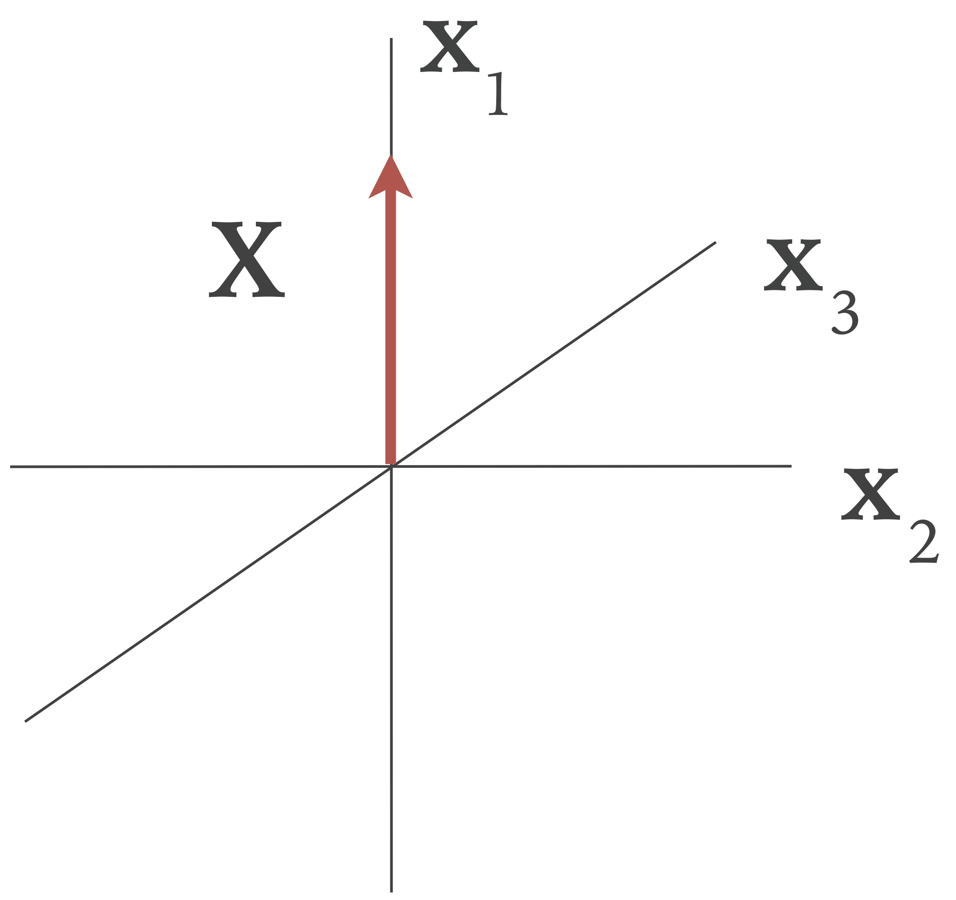


$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

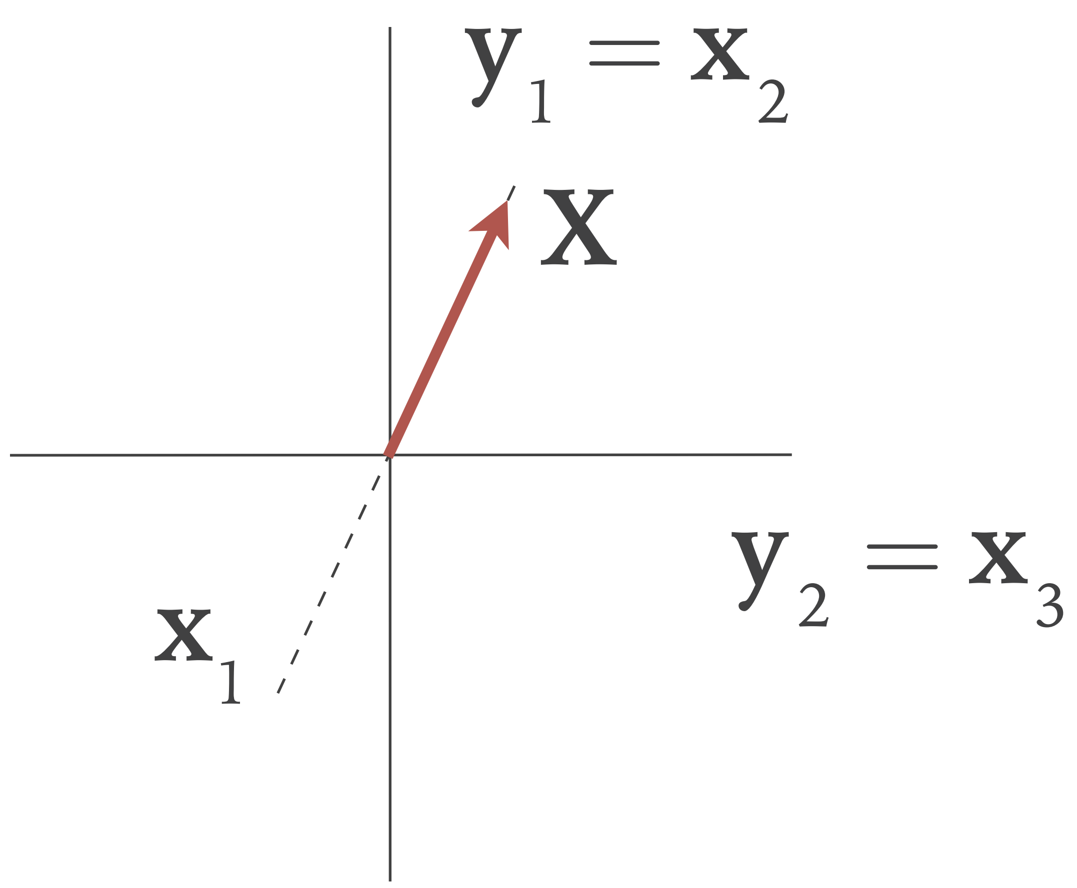


Example 1-sparse case

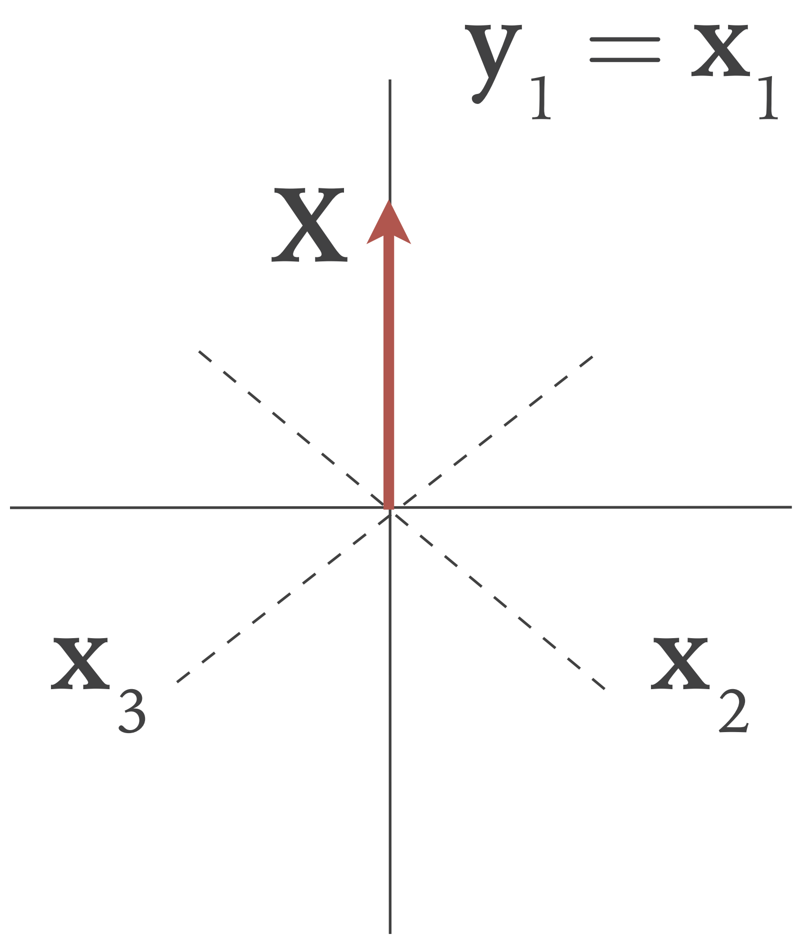
- Some good measurement matrices



$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix}$$



$$\mathbf{P} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$$



Embedding viewpoint

- This is similar to the subspace/manifold methods
  - Find low-rank projection that preserves cluster characteristics
- Special case: Random projection
  - For n points in p-dims there exists a q-dim projection that preserves distances by a factor of $1+\varepsilon$, $q \geq O(\varepsilon–2 \log n)$

Restricted Isometry Property (RIP)

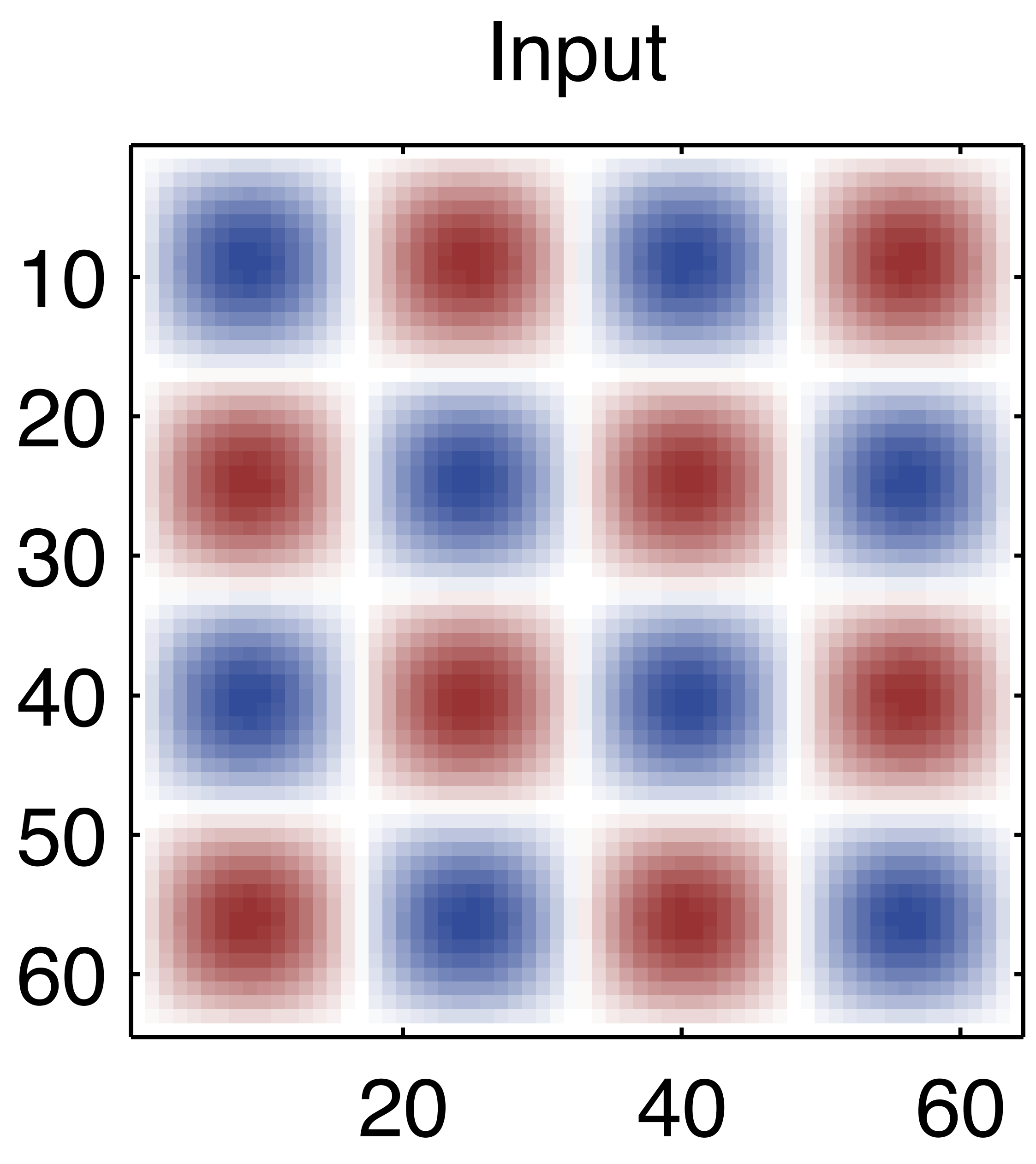
- A measurement matrix $\mathbf{P}$ has order-K RIP if:

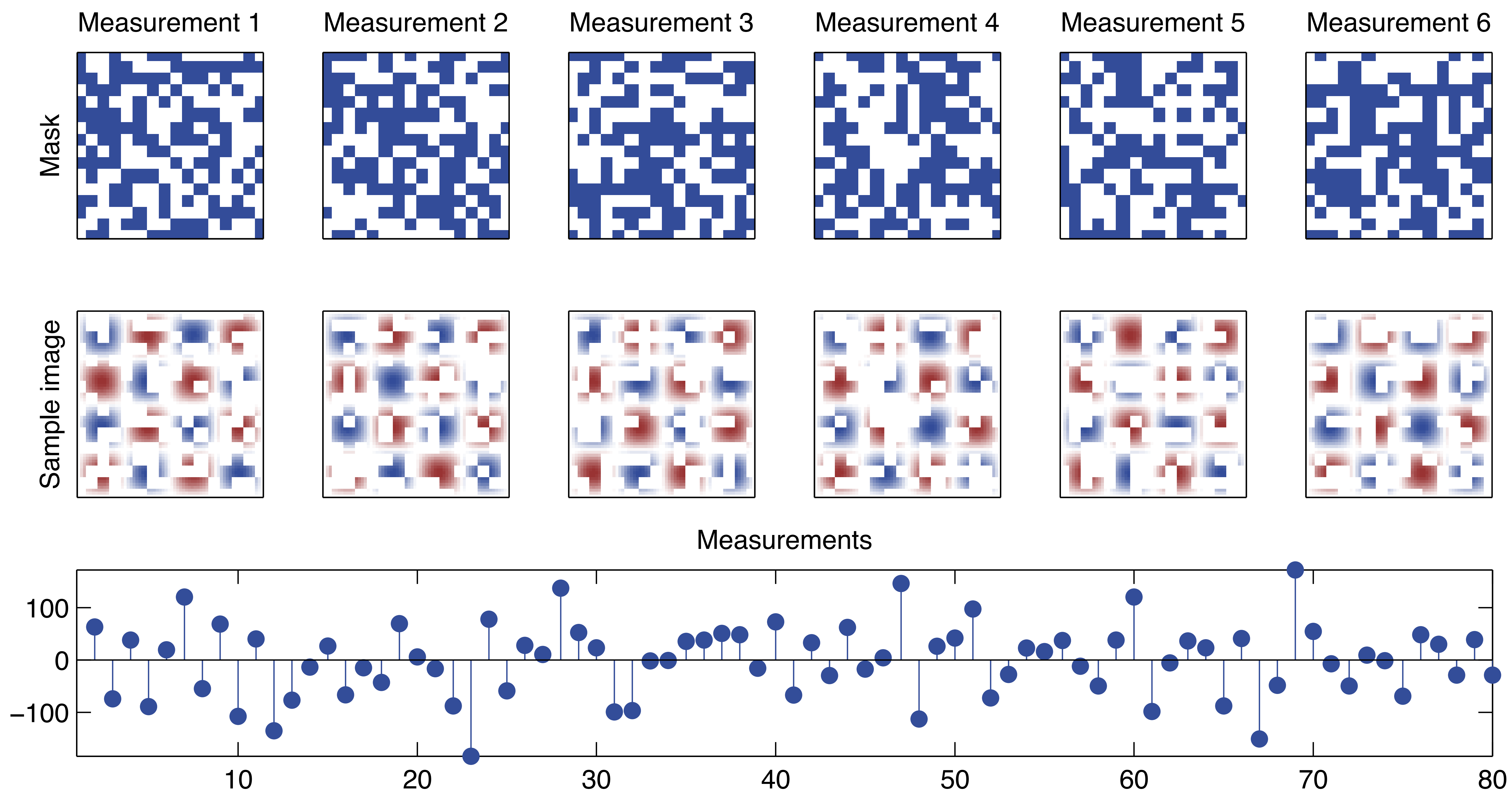
$$\left(1-\delta_K\right) \leq \frac{\left\|\mathbf{P}\cdot\mathbf{x}\right\|_2^2}{\left\|\mathbf{x}\right\|_2^2} \leq \left(1+\delta_K\right),\ \forall\ K\text{-sparse } \mathbf{x}$$

- How do we check? Difficult …
  - But we have some good choices
    - Gaussian, Bernoulli, Fourier, etc.

An simple compressed sensing case

- Simple 64 × 64 input
  - Take a set of masked measurements, store only average value
  - Measure using: $y_i = \mathbf{p}_i^\top \cdot \text{vec}\left(\mathbf{x}\right), \quad \mathbf{p}_i \in \{0,1\}$





Resolving via the DCT

- Model is:

$$\mathbf{y} = \mathbf{P}^\top \cdot \mathbf{x} = \mathbf{P}^\top \cdot \left(\mathbf{C}^\top \cdot \mathbf{z}\right)$$

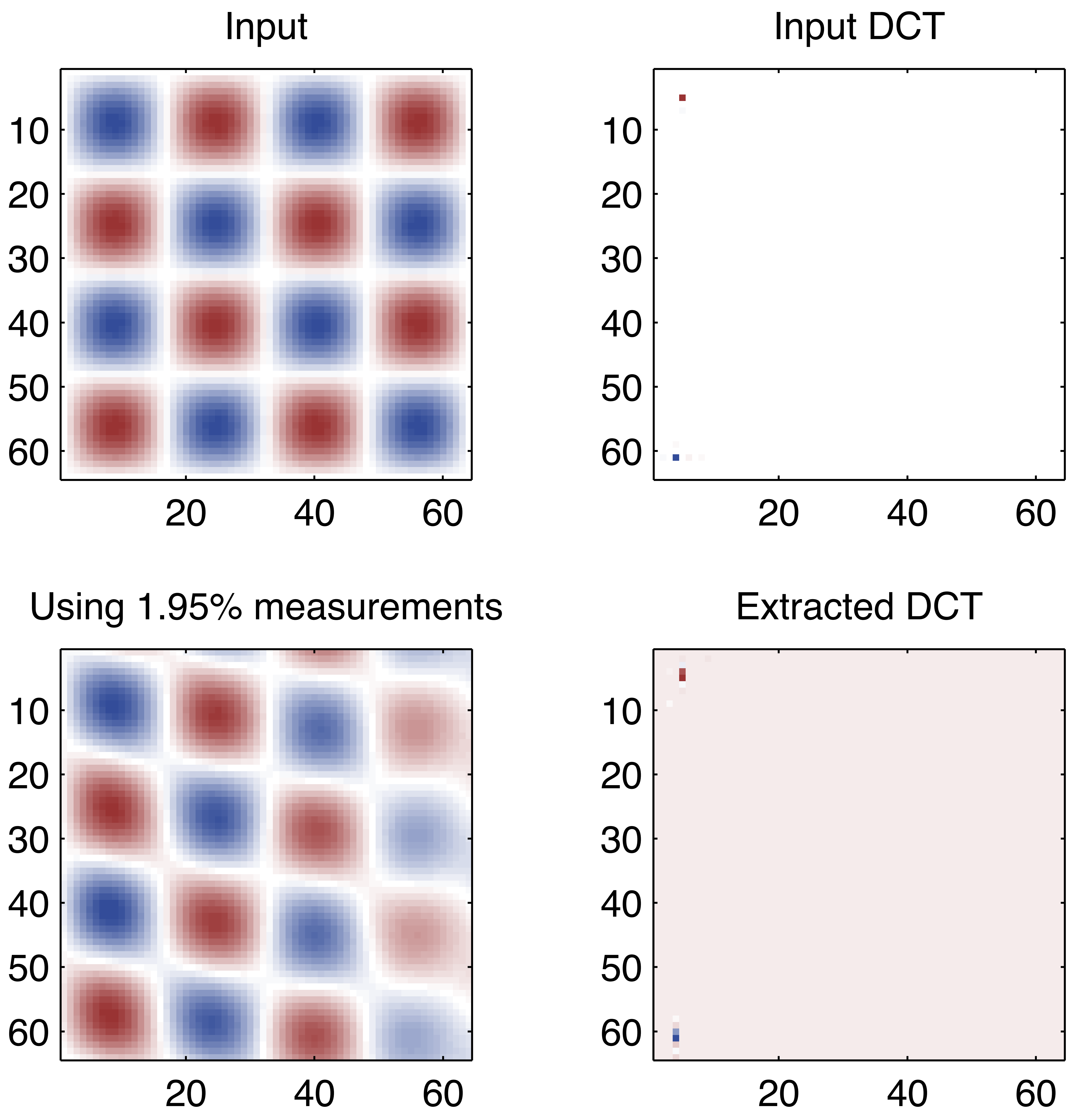
  - Assume sparsity in $\mathbf{z}$
- Resolve:

$$\min \left\|\mathbf{z}\right\|_1 + \left\|\mathbf{y} - \mathbf{P}^\top \cdot \mathbf{C}^\top \cdot \mathbf{z}\right\|$$

- Reconstruct:

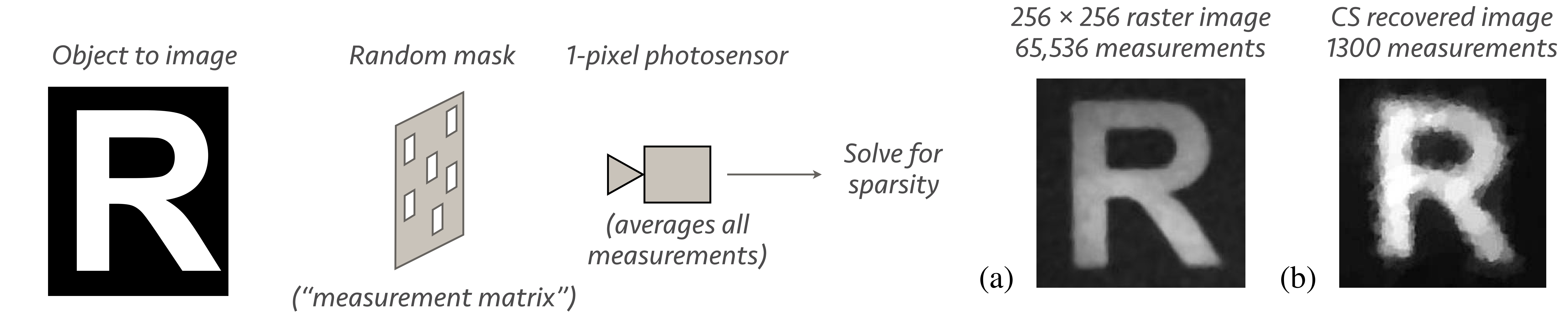
$$\hat{\mathbf{x}} = \mathbf{C}^\top \cdot \mathbf{z}$$

  - Using ~2% of samples!



Putting compressive sensing to work

- The single-pixel camera
  - Measure image intensity using multiple random masks
  - Reconstruct assuming sparsity
    - In some domain …

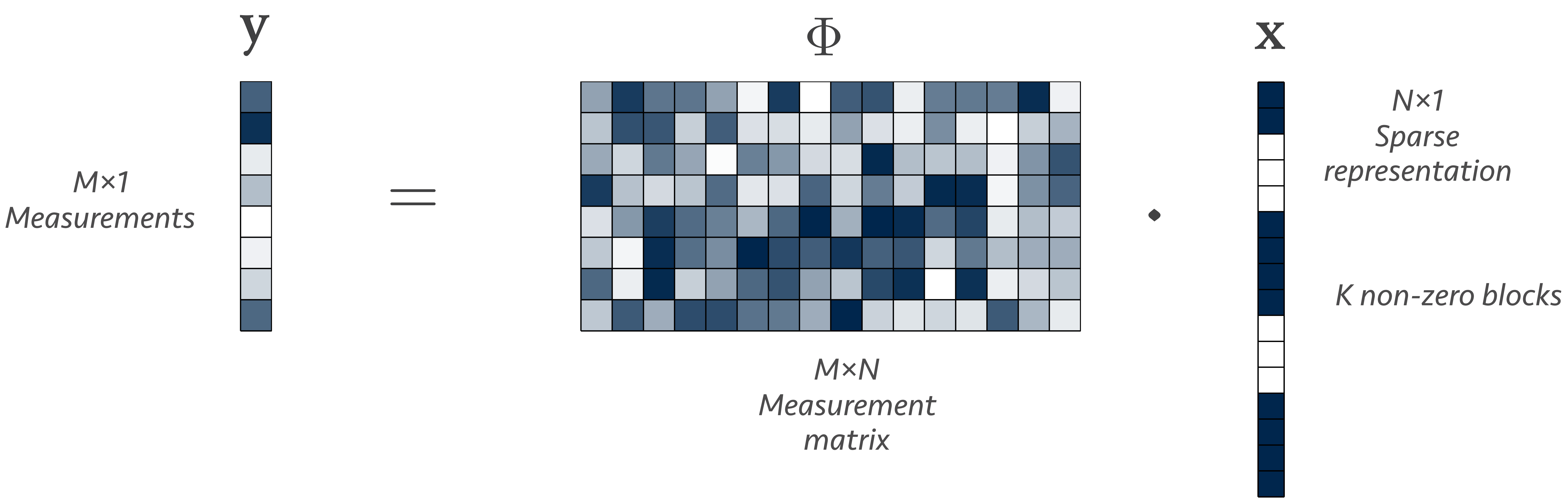


Other kinds of sparsity

- Up to now we talked about a simple form of sparsity
  - Sparsity over all coefficients separately
- Some problems need more elaborate definitions
  - e.g. block structure, joint structure, temporal structure, etc.

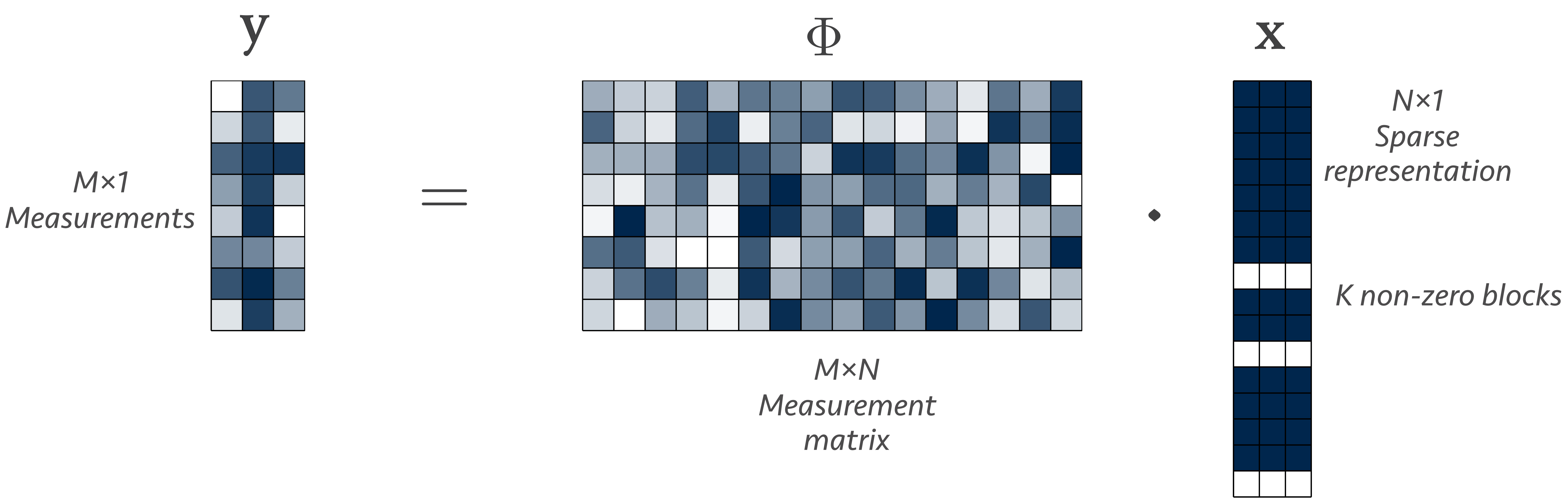
Block sparsity

- Have sparsity appear in non-overlapping blocks
  - Useful for some imaging operations



Joint sparsity

- Obtain multiple solutions with the same sparsity
  - Useful for multimodal/multichannel data



Learning and sparsity

- How about other models?
  - Add sparsity as an extra "regularizer"
- Sparse decompositions
  - Sparse NMF, sparse PCA, etc …
- ICA is already (sort of) sparse!

Recap

- Sparsity and ℓ_p norms
  - Different definition of sparsity
- Minimum-ℓ_1 coefficient algorithms
  - Linear programming
  - Greedy methods
- Compressive sensing and random projections
  - 1-pixel camera

Reading material

- Compressive sensing
  - http://dsp.rice.edu/sites/dsp.rice.edu/files/cs/baraniukCSlecture07.pdf
- Compressive Sensing page
  - http://dsp.rice.edu/cs
- Experiments with Random Projections
  - http://dimacs.rutgers.edu/Research/MMS/PAPERS/rp.pdf

Next lecture

- Deep learning!
  - aka neural nets v2.0

- Also Problem Set 4 is out
  - Optional, due at last day of classes
  - Use it to perk up your grade if you need to
    - Would also help if your final project is floundering