Sublinear Algorithms for Compressed Sensing

Joel A. Tropp

Department of Mathematics
The University of Michigan
jtropp@umich.edu

Joint work with
Anna C. Gilbert
Martin J. Strauss
Roman Vershynin

Research supported in part by NSF DMS Grants #0503299, #0354600, #0401032

Act I: Motivation

What is Compressed Sensing?

Compressed Sensing is the idea that sparse signals can be reconstructed from a small number of nonadaptive linear measurements.

Example:

- Draw $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from the standard normal distribution on $\mathbb{R}^d$
- Let $\boldsymbol{s}$ be any m-sparse signal in $\mathbb{R}^d$
- Collect measurements $\langle \boldsymbol{s},\ \boldsymbol{x}_1 \rangle, \ldots, \langle \boldsymbol{s},\ \boldsymbol{x}_N \rangle$
- Solve

$$\min_{\widehat{\boldsymbol{s}}} \|\widehat{\boldsymbol{s}}\|_1 \quad \text{subject to} \quad \langle \widehat{\boldsymbol{s}},\ \boldsymbol{x}_n \rangle = \langle \boldsymbol{s},\ \boldsymbol{x}_n \rangle \quad \text{for } n = 1, 2, \ldots, N$$

- **Result:** $\widehat{\boldsymbol{s}} = \boldsymbol{s}$ provided that $N \sim m \log d$

References: Candès–Tao 2004, Donoho 2004

Application Areas

Compressed Sensing reverses the usual paradigm in source coding:

The encoder is resource poor. The decoder is resource rich.

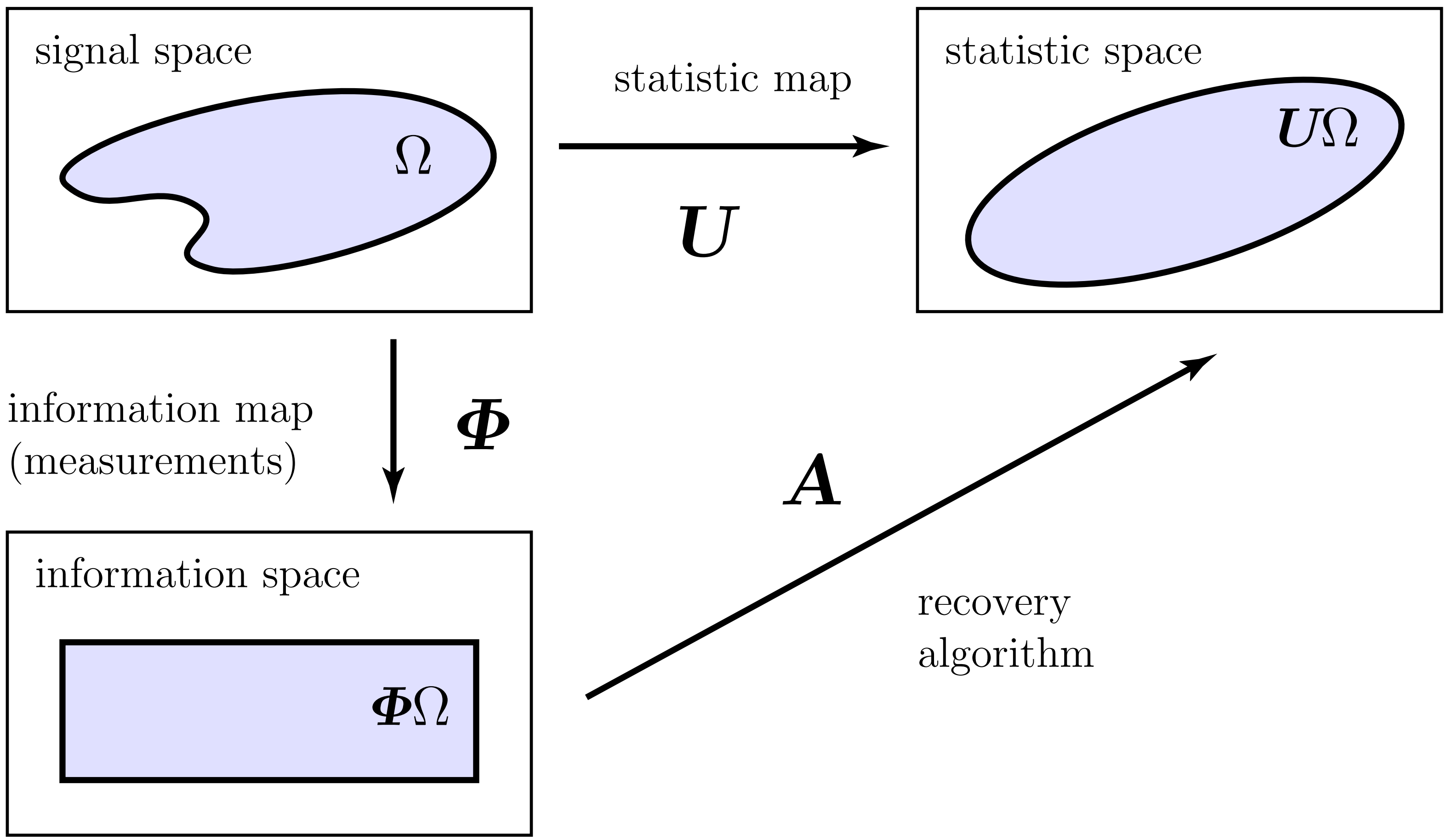
1. Medical imaging
2. Sensor networks

Compressed Sensing is also relevant in case

signals are presented in streaming form or are too long to instantiate.

1. Inventory updates
2. Network traffic analysis

Optimal Recovery

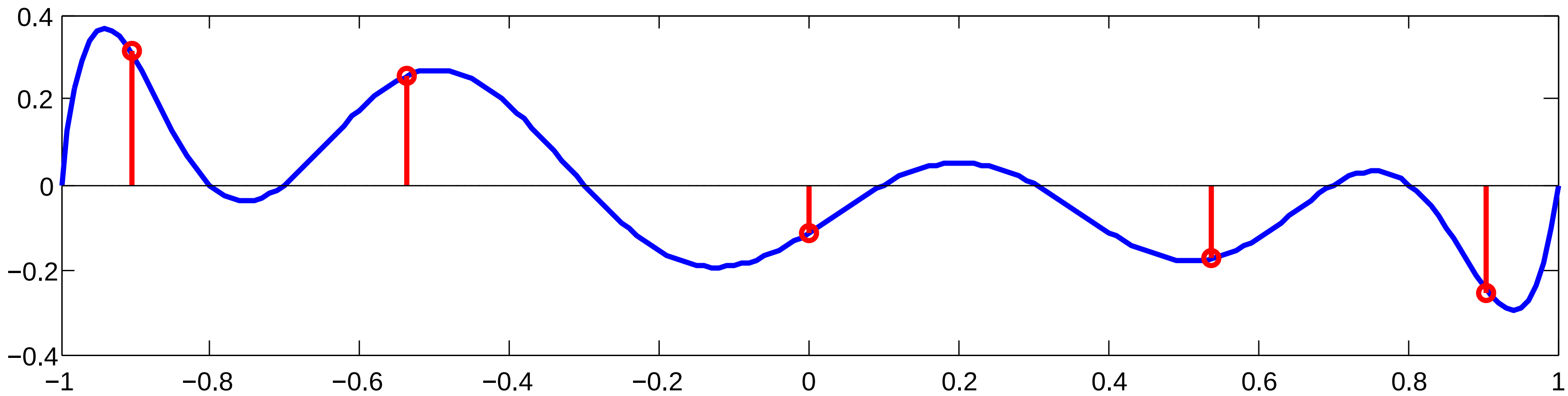


Reference: Golomb–Weinberger 1959

Questions. . .

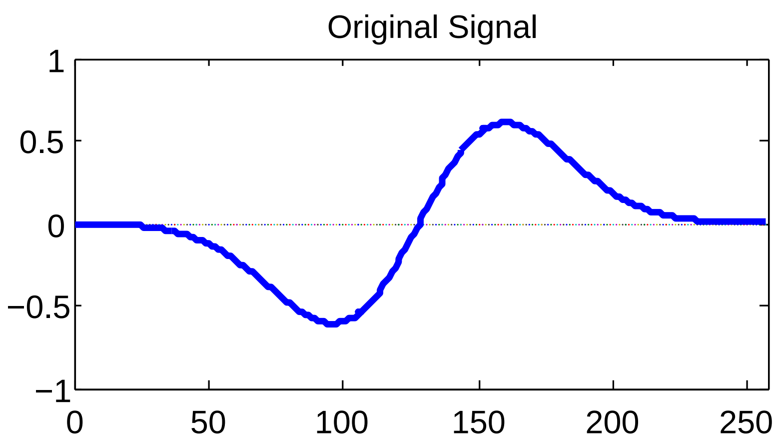
- What signal class are we interested in?
- What statistic are we trying to compute?
- How much nonadaptive information is necessary to do so?
- What type of information? Point samples? Inner products?
- Deterministic or random information?
- How much storage does the measurement operator require?
- How much computation time, space does the algorithm use?

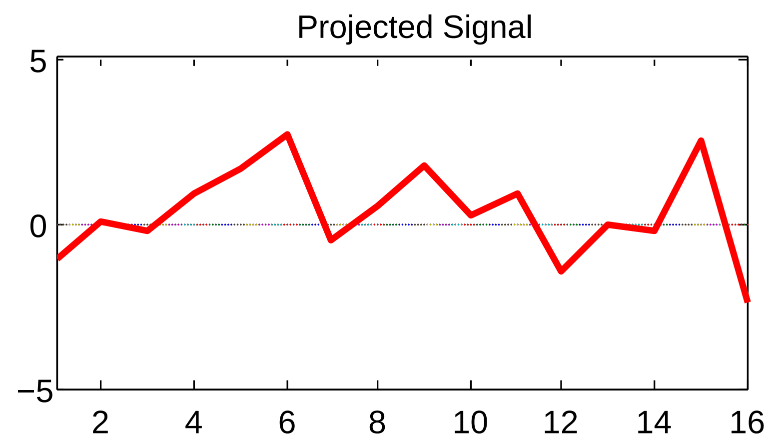
Example 1: Gaussian Quadrature



- **Signals:** Polynomials of degree at most $(2n-1)$
- **Statistic:** The integral $\int_{-1}^{1} p(t)\,\mathrm{d}t$
- **Measurements:** Function values at n fixed points $t_1, \ldots, t_n$
- **Algorithm:** The linear quadrature rule $\sum_{i=1}^{n} w_i\, p(t_i)$
- **Result:** Perfect calculation of the integral
- **Note:** The polynomial cannot be reconstructed from the samples!

Example 2: Norm Approximation

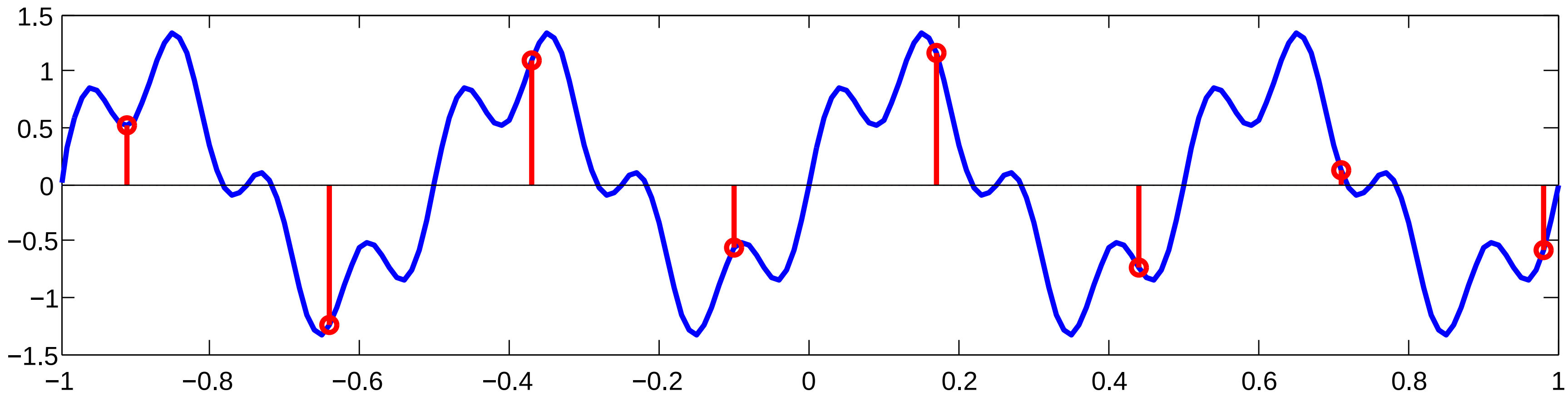




- **Signals:** All signals in a d-dimensional Euclidean space
- **Statistic:** The ℓ_2 norm of the signal
- **Measurements:** A *random* projection onto $O(1/\varepsilon^2\delta)$ dimensions
- **Algorithm:** Compute the (scaled) ℓ_2 norm of the projected signal
- **Result:** Norm correct within a factor $(1 \pm \varepsilon)$ with probability $(1 - \delta)$
- **Note:** The ambient dimension d plays no role!

Reference: Johnson–Lindenstrauss 1984

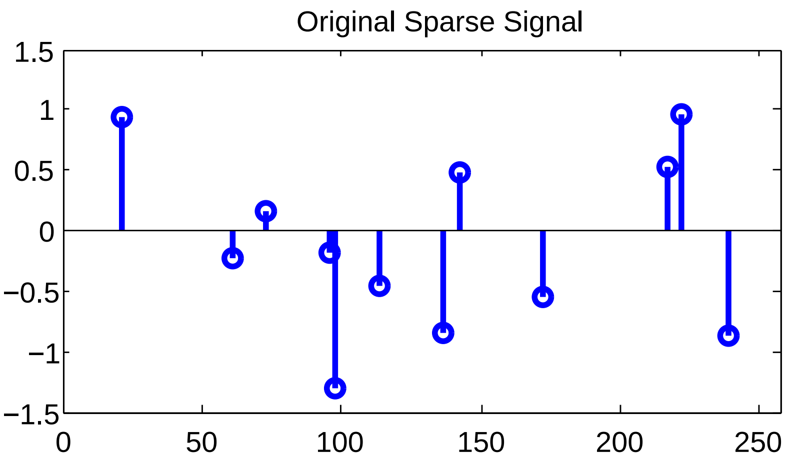
Example 3: Fourier Sampling

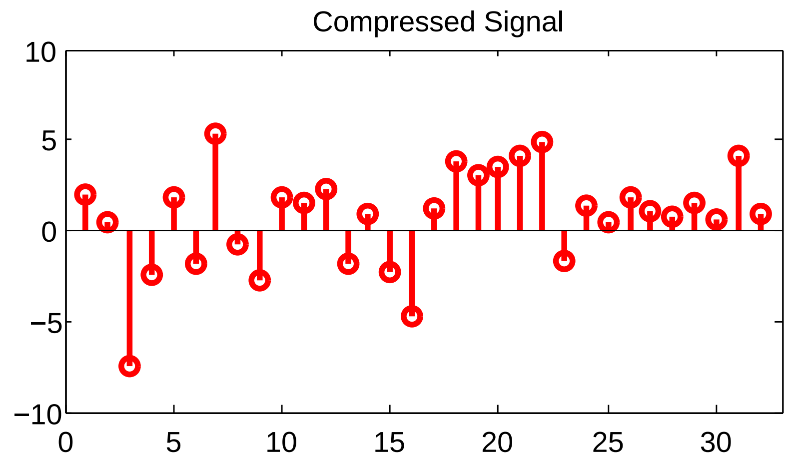


- **Signals:** All vectors in a d-dimensional Euclidean space
- **Statistic:** The largest m Fourier coefficients
- **Measurements:** $O(m \log^2 d/\varepsilon^2\delta)$ structured random point samples
- **Algorithm:** A *sublinear, small space* greedy pursuit
- **Result:** An m-term trig polynomial with ℓ_2 error at most $(1+\varepsilon)$ times optimal with probability $(1-\delta)$

References: Gilbert et al. 2002, 2005

Example 4: Compressed Sensing

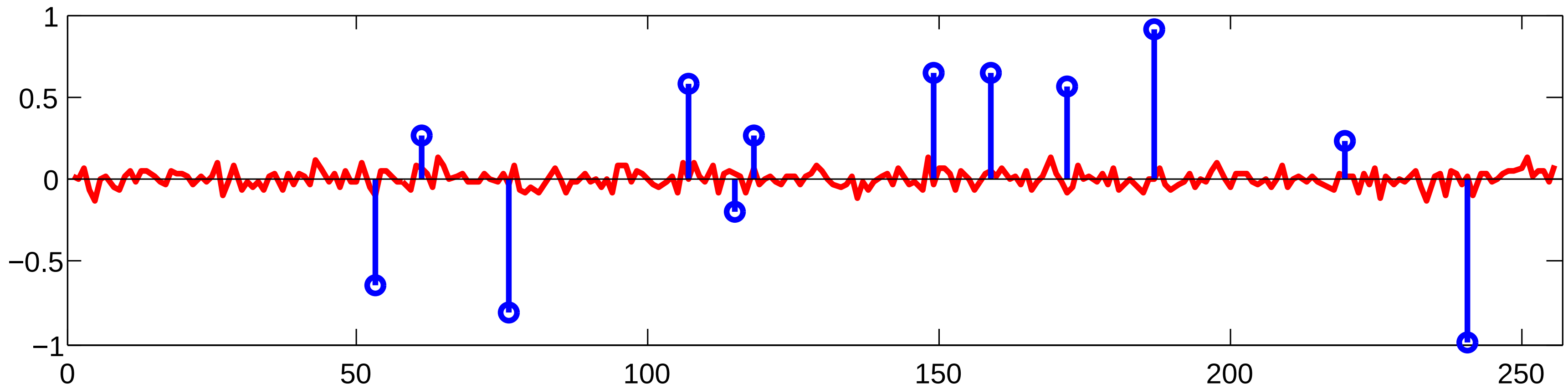




- **Signals:** All m-sparse vectors in d dimensions
- **Statistic:** Locations and values of the m spikes
- **Measurements:** A fixed projection onto $O(m \log d)$ dimensions
- **Algorithm:** ℓ_1 minimization
- **Result:** Exact recovery of the signal
- **Note:** Computation time/space is *polynomial in d*

References: Candès et al. 2004, Donoho 2004, . . .

Our Goal



- **Signals:** All noisy m-sparse vectors in d dimensions
- **Statistic:** Locations and values of the m spikes

- **Measurement Goal:** $m \operatorname{polylog} d$ fixed measurements
- **Algorithmic Goal:** Computation time $m \operatorname{polylog} d$
- **Error Goal:** Error proportional to the optimal m-term error

Act II: Measurements

Locating One Spike

- Suppose the signal contains one spike and no noise
- $\log_2 d$ bit tests will identify its location, e.g.,

$$\boldsymbol{B}_1 \boldsymbol{s} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{matrix} \text{MSB} \\ \\ \text{LSB} \end{matrix}$$

bit-test matrix · signal = location in binary

Estimating One Spike

- Suppose the signal contains one spike and no noise
- Its size can be determined from any nonzero bit test, e.g.,

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0.8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.8 \\ 0 \end{bmatrix}$$

Bit Tests, with Noise

- Suppose the signal contains one spike *plus noise*
- The spike location and value can be estimated by comparing bit tests with their complements:

Bit = 1

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.2 \\ 1.0 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.9 \\ -0.3 \end{bmatrix}$$

Bit = 0

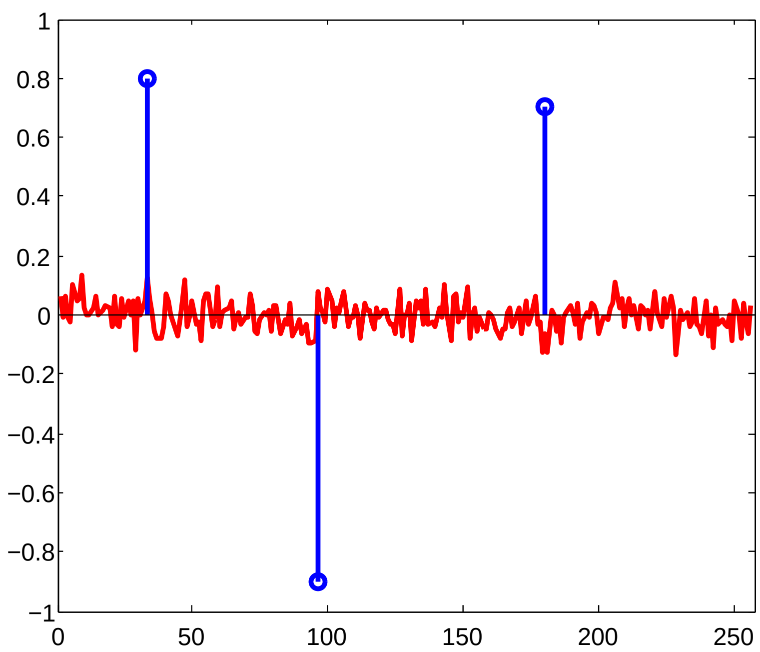
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.2 \\ 1.0 \\ -0.1 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 1.1 \end{bmatrix}$$

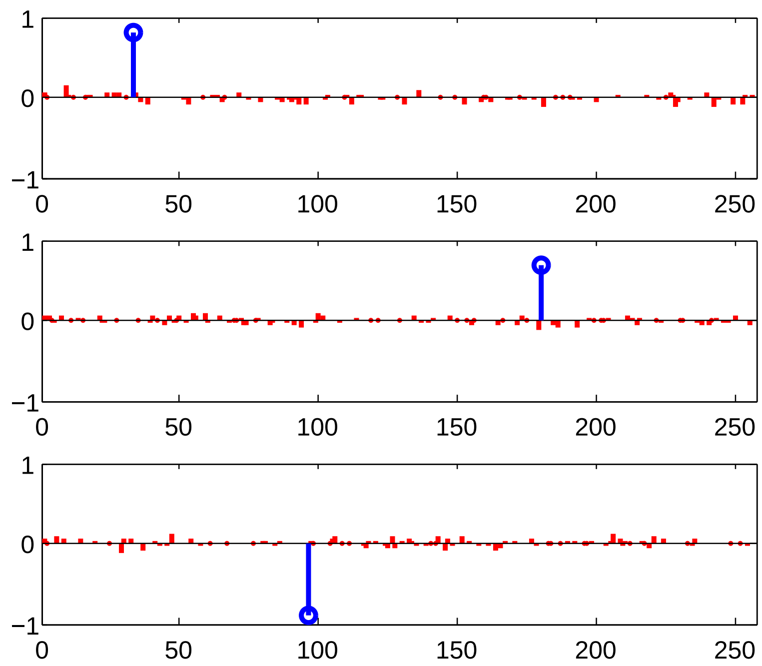
- $|0.9| > |-0.1|$ implies the location MSB = 1
- $|1.1| > |-0.3|$ implies the location LSB = 0

- Using the LSB, we estimate the value as 1.1

Isolating Spikes

- To use bit tests, the measurements need to isolate the m spikes
- Assign each of d signal positions to one of $O(m)$ different subsets, uniformly at random
- Apply bit tests to each subset





Isolation, Matrix Form

- A partition of signal positions into subsets can be encoded as a 0–1 matrix, e.g.,

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- The first row shows which positions are assigned to the first subset, etc.

- Partitioning the signal T times can be viewed as a block matrix:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \hline \boldsymbol{A}_2 \\ \hline \vdots \\ \hline \boldsymbol{A}_T \end{bmatrix}$$

The Measurement Operator

- The measurement operator $\boldsymbol{\Phi}$ consists of an isolation matrix $\boldsymbol{A}$ and a bit-test matrix $\boldsymbol{B}$. It acts as

$$\boldsymbol{\Phi s} = \begin{bmatrix} \boldsymbol{A}_1 \\ \hline \boldsymbol{A}_2 \\ \hline \vdots \\ \hline \boldsymbol{A}_T \end{bmatrix} \cdot \operatorname{diag} \boldsymbol{s} \cdot \begin{bmatrix} \boldsymbol{B}_0 \\ \hline \boldsymbol{B}_1 \end{bmatrix}^T$$

- Each $\boldsymbol{A}_t$ randomly partitions the signal into $N = O(m)$ subsets
- The number of trials $T = O(\log m \cdot \log d)$

- Each row of $\boldsymbol{V} = \boldsymbol{\Phi s}$ contains bit tests applied to one subset

- $\boldsymbol{V}$ contains $O(m \log m \log^2 d)$ measurements

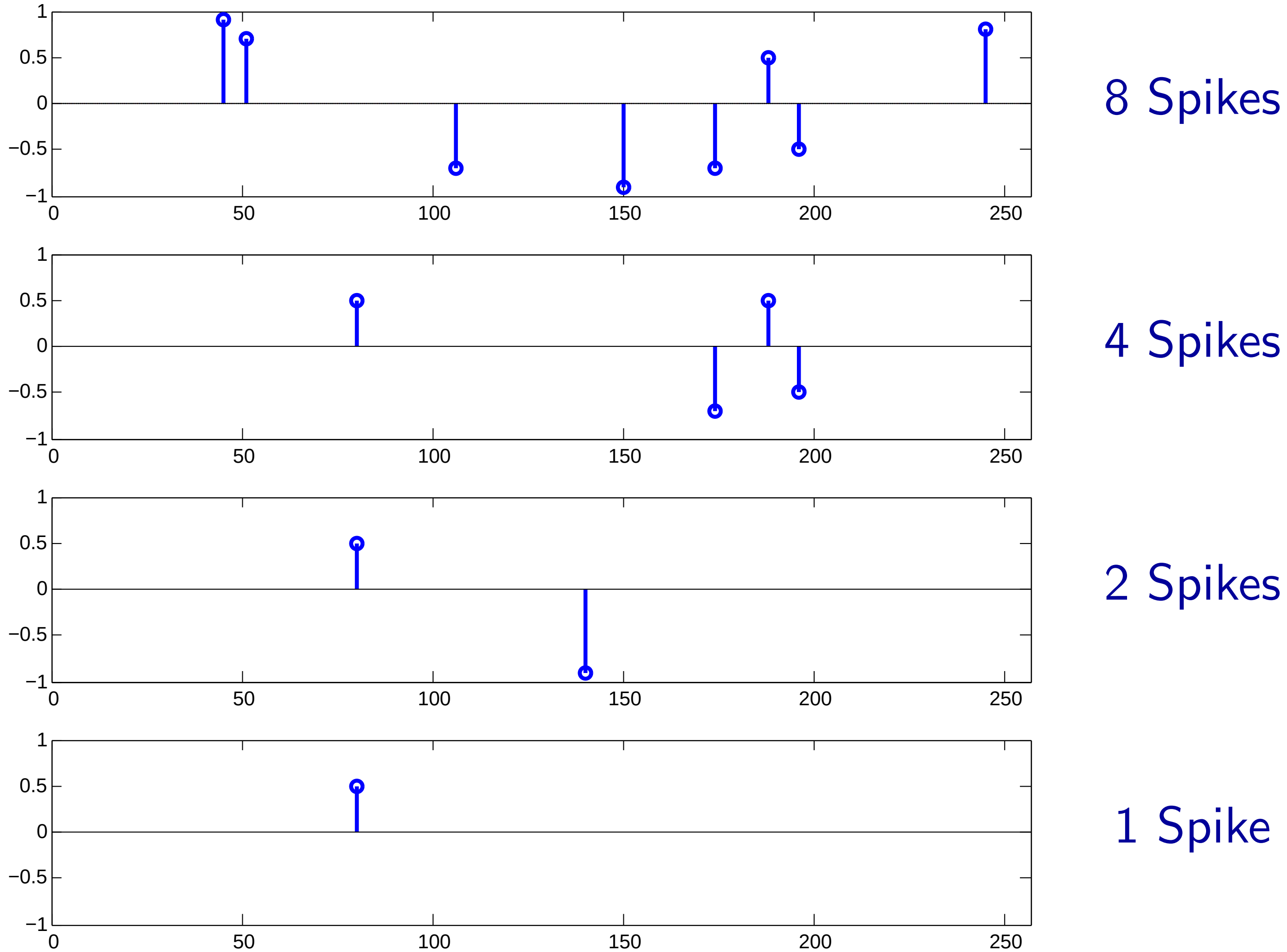
Properties of the Measurement System

For any collection of m spikes and any noise,

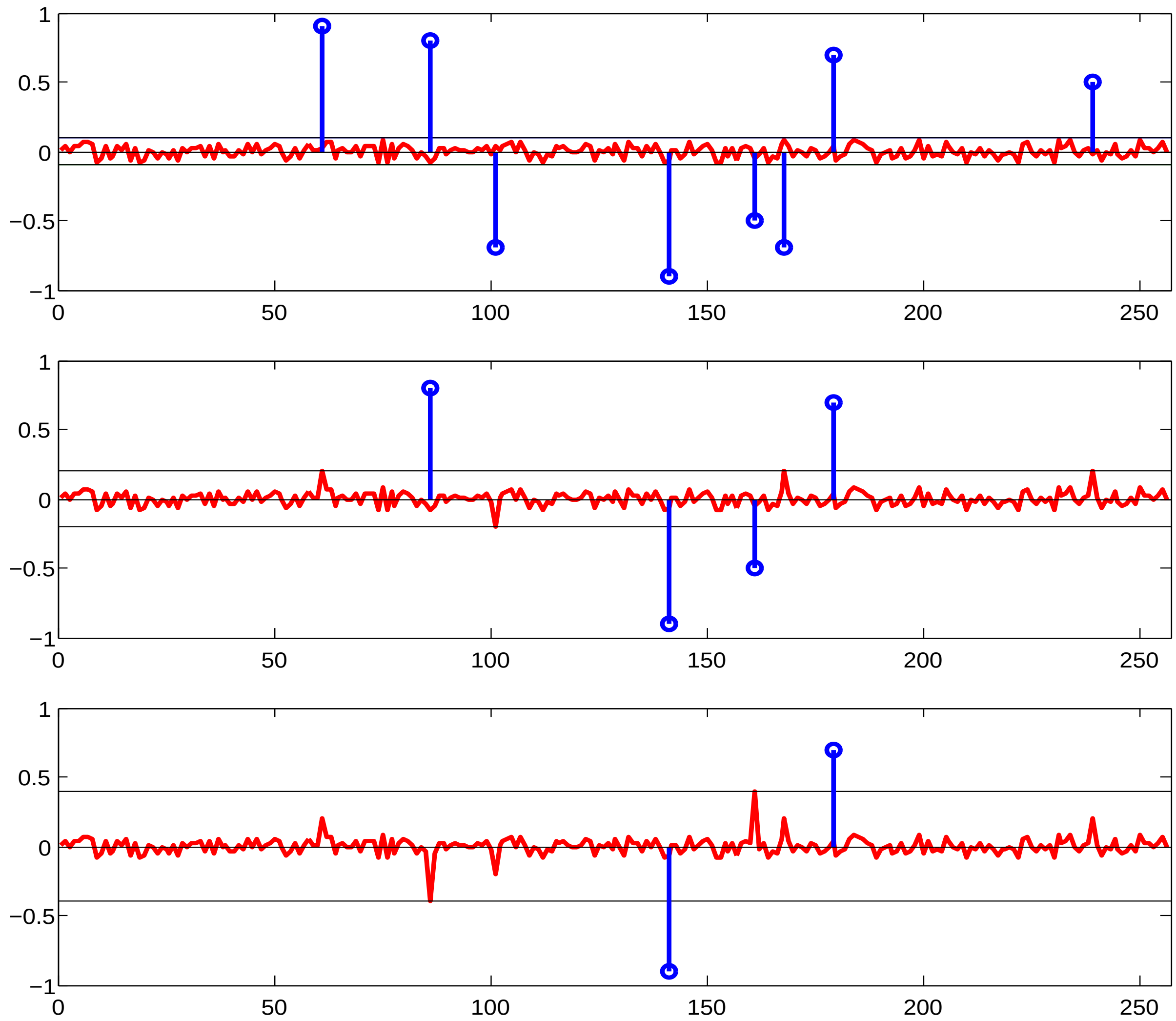
- In most of the trials, most of the spikes are isolated
- In each trial, few subsets contain an unusual amount of noise
- The measurements have other technical properties that are necessary to make the algorithm work
- The technical claims are difficult to establish

Act III: Algorithm

Geometric Progress



Geometric Progress, with Noise



8 Spikes
Noise Level 0.1

4 Spikes
Noise Level 0.2

2 Spikes
Noise Level 0.4

Chaining Pursuit

```
Inputs:  Number of spikes m, data V, isolation matrix A
Output:  A list of ≤ 3m spike locations and values

For each pass k = 0, 1, ..., log_2 m:
     For each trial t = 1, 2, ..., T:

         For each measurement n = 1, ..., N
             Use bit tests to identify the spike position
             Use a bit test to estimate the spike magnitude

         Retain m/2^k distinct spikes with largest values

     Retain spike positions that appear in 2/3 of trials
     Estimate final spike magnitudes using medians

     Encode the spikes using the measurement operator
     Subtract the encoded spikes from the data matrix
```

Algorithmic Costs

Storage

The storage cost is dominated by the measurement system $\mathbf{\Phi}$

Total storage cost: $O(d \log m \log^2 d)$

Time

The time cost is dominated by the sort in the second loop

Total time cost: $O(m \log m \log^2 d)$

Theoretical Guarantee

- Let $\mathbf{\Phi}$ be a random measurement operator with

$$N = O(m) \qquad \text{and} \qquad T = O(\log m \cdot \log d)$$

- Except with probability $d^{-O(1)}$, the following theorem holds

Theorem 1. [Chaining Pursuit] *Let $\boldsymbol{s}$ be any d-dimensional signal, and let $\boldsymbol{s}_m$ be its best m-sparse approximation. On inputs m and $\boldsymbol{V} = \mathbf{\Phi}\boldsymbol{s}$, Chaining Pursuit produces a signal estimate $\widehat{\boldsymbol{s}}$ consisting of $3m$ spikes or fewer. This signal estimate satisfies*

$$\left\| \boldsymbol{s} - \widehat{\boldsymbol{s}} \right\|_1 \leq \mathrm{C} \log m \left\| \boldsymbol{s} - \boldsymbol{s}_m \right\|_1.$$

In particular, if $\boldsymbol{s}_m = \boldsymbol{s}$, then also $\widehat{\boldsymbol{s}} = \boldsymbol{s}$.

Notes on the Theorem

Theorem 1 can be refined in several ways:

- The number of spikes in the output can be controlled
- The proof actually provides a sharper error bound:

$$\left\| \boldsymbol{s} - \widehat{\boldsymbol{s}} \right\|_{\text{weak-1}} \leq \mathrm{C}' \left\| \boldsymbol{s} - \boldsymbol{s}_m \right\|_1 .$$

- The total number of measurements can be reduced to $O(m \log^2 d)$ with a more complicated measurement system

Comparison with Other Approaches

	FourierSamp	Chaining	ℓ_1, Gaussian	ℓ_1, Fourier
Error Bound	ℓ_2	weak ℓ_1/ℓ_1	ℓ_2/ℓ_1	ℓ_2/ℓ_1
Uniform	No	Yes	Yes	Yes
# Measures	$m\log^2 d$	$m\log^2 d$	$m\log(d/m)$	$m\log^4 d$
Storage cost	$m\log^2 d$	$d\log^2 d$	$md\log(d/m)$	$m\log^5 d$
Decode time	$m\log^2 d$	$m\log^2 d$	d^3	$d\log d$ (empir.)

Note: For legibility, the big-O notation and some factors of $\log m$ are suppressed.

References: Gilbert et al. 2002, 2005; Candès et al. 2004, 2005; Donoho 2004, 2005; Rudelson–Vershynin 2006; Cohen et al. 2006

Act IV: Experiments

Example: Spikes, without noise

```
Phi = GenerateMeasurements( 1024, 16, 4 );
s = [ four randomly located spikes ];
V = EncodeSignal( s, Phi );
hat = ChainingPursuit( 4, V, Phi );
```

```
  s =                                   hat =

   (167,1)       0.8670                  (167,1)       0.8670
   (230,1)       0.5017                  (230,1)       0.5017
   (563,1)       0.5547                  (563,1)       0.5547
   (764,1)       0.9342                  (764,1)       0.9342
```

```
Elapsed time is 0.044399 seconds.
```

Example: Signals in weak ℓ_1

```
s =

     1.0000
     0.5000
     0.3333
    -0.2500
     0.2000
    -0.1667
    -0.1429
    -0.1250
        ...
```

```
hat =

   (1,1)        0.9629
   (2,1)        0.5040
   (3,1)        0.3826
   (4,1)       -0.1944
   (5,1)        0.1892
   (6,1)       -0.1636
```

```
Elapsed time is 0.087329 seconds.
Error in approximation, l1 norm: 5.219
Error in best 8-term approximation, l1 norm: 4.7913
```

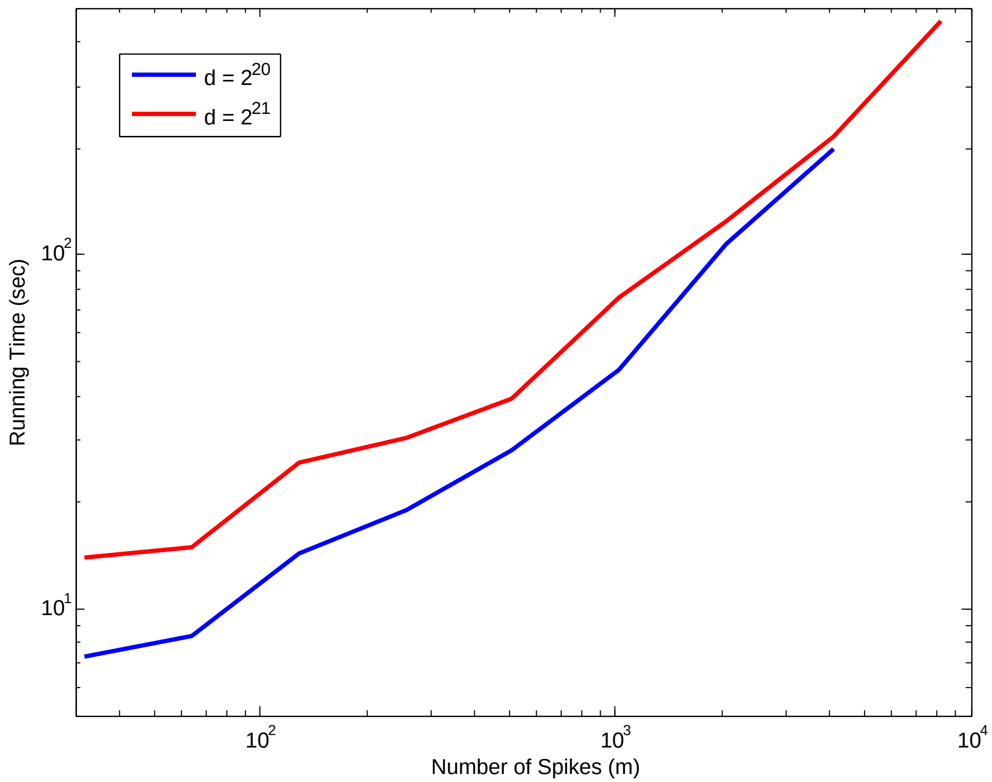
Scalability

Even in Matlab, large problems are within reach...

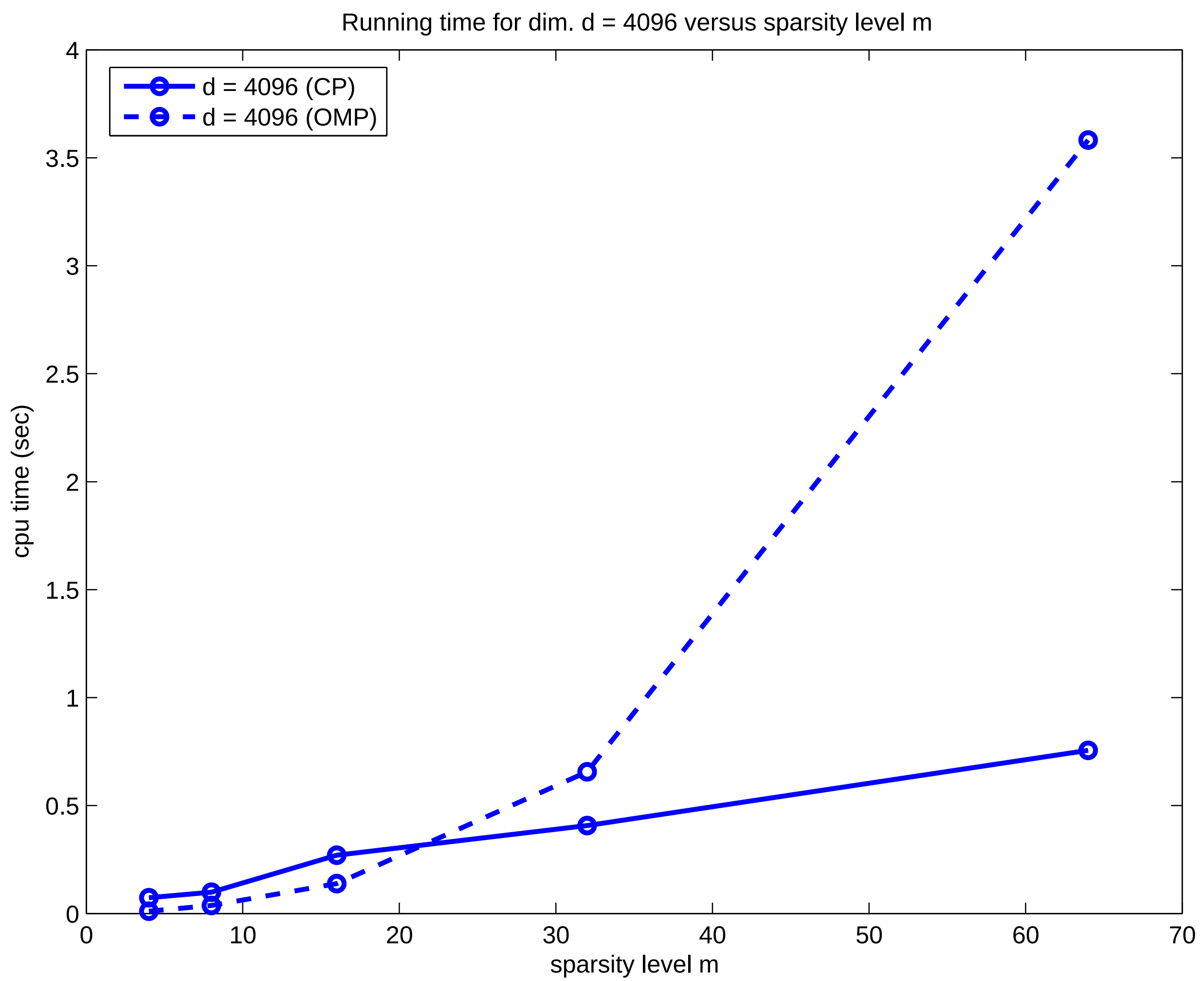
Sample Running Times for Chaining Pursuit (sec)

m	$d = 2^{20}$	$d = 2^{21}$
2^5	7.3	14.0
2^6	8.4	15.0
2^7	14.4	25.8
2^8	19.1	30.5
2^9	28.1	39.1
2^{10}	47.5	76.0
2^{11}	108.0	123.9
2^{12}	200.2	216.4
2^{13}	—	460.2

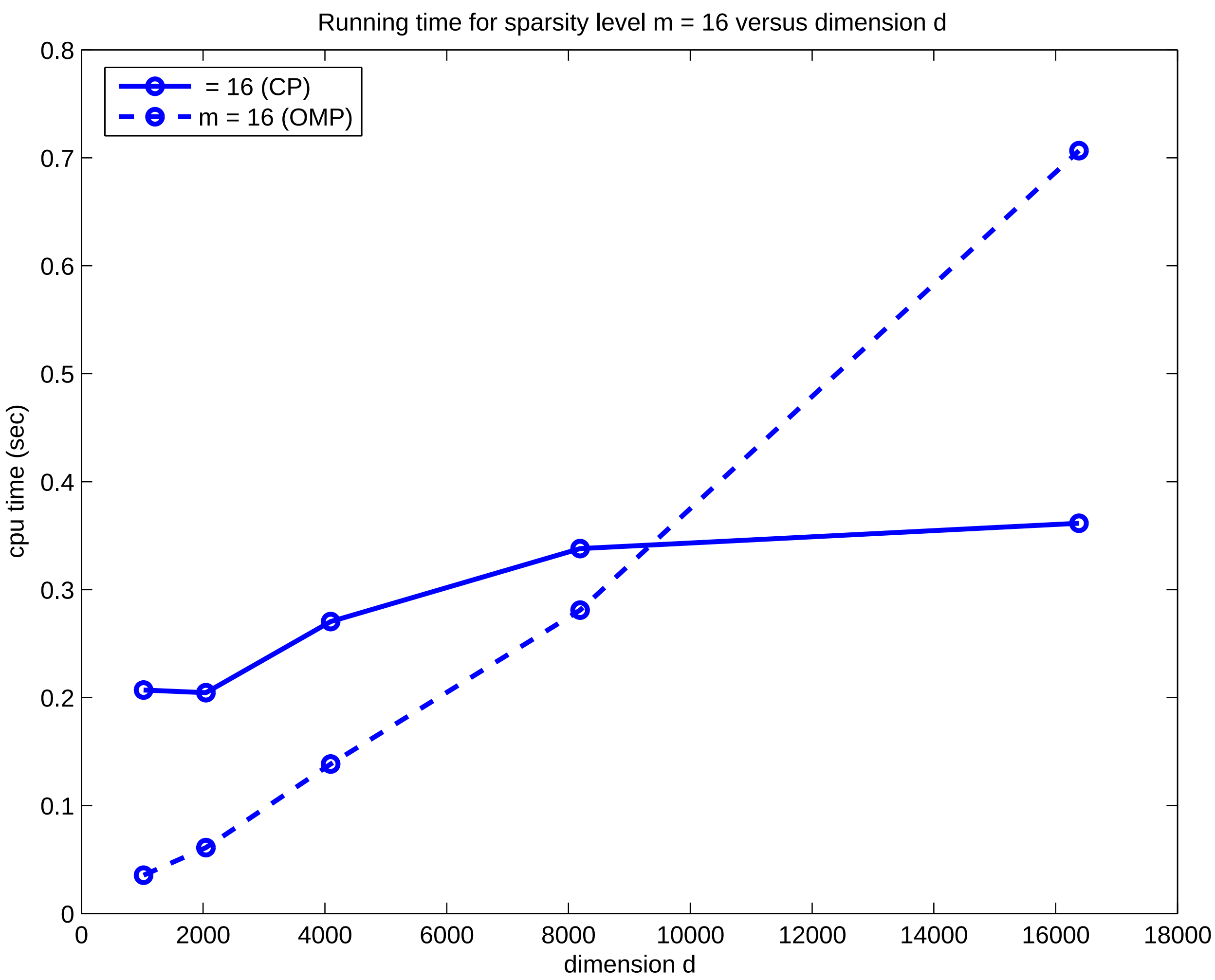
Note: $N = 4m$, $T = 32$



Comparative Timings: $d = 4096$



Comparative Timings: $m = 16$



Act V: Extensions

Shortcomings of Chaining Pursuit

1. Storage is proportional to the length of the signal
2. Cannot make the approximation error arbitrarily close to optimal
3. Error bound is in terms of the ℓ_1 norm

K2 Pursuit

K2 Pursuit improves on Chaining Pursuit in several regards. . .

- **Measurements:** $m\,\mathrm{poly}(\varepsilon^{-1}\log d)$
- **Storage:** $m\,\mathrm{poly}(\varepsilon^{-1}\log d)$
- **Error bound:**

$$\|\,\boldsymbol{s} - \widehat{\boldsymbol{s}}\,\|_2 \leq \|\boldsymbol{s} - \boldsymbol{s}_m\|_2 + \frac{\varepsilon}{\sqrt{m}}\,\|\boldsymbol{s} - \boldsymbol{s}_m\|_1\,.$$

- **Time:** $m^2\,\mathrm{poly}(\varepsilon^{-1}\log d)$

We're working to reduce the computation time...

To learn more...

Web: `http://www.umich.edu/~jtropp`

E-mail: `jtropp@umich.edu`

- Matlab code for Chaining Pursuit is freely available!
- ACG, MJS, JAT and RV, "Sublinear approximation of compressible signals," to appear, SPIE IIM 2006
- —, "Algorithmic dimension reduction in the ℓ_1 norm for sparse vectors," submitted April 2006
- JAT and ACG, "Signal recovery from partial information via Orthogonal Matching Pursuit," submitted April 2005
- JAT and Rice DSP, "Random filters for compressive signal acquisition and reconstruction," to appear, ICASSP 2006
- S. Sra and JAT, "Row-action methods for Compressed Sensing," to appear, ICASSP 2006