On Compressed Sensing and on to Sparsity

3.1

$$f(x) = \sin x + 3\sin 3x + 4\sin 7x \in L^2([0,\pi])$$

$$= \sum_{n=1}^{\infty} a_n \sin nx \qquad \frac{\sin kx}{\ell}$$

$$\hat{f}(x) = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0 \\ 4 \\ 0 \\ \vdots \end{pmatrix}$$

$$; \quad f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

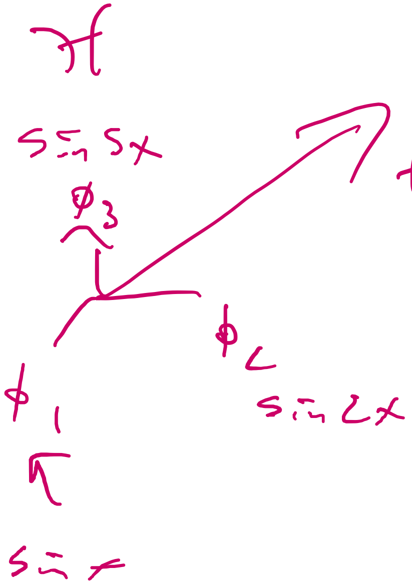
$$+ 3\left(3x - \frac{(3x)^3}{3!} + \frac{(3x)^5}{5!} - \cdots\right)$$

$$+ 4\left(7x - \frac{(7x)^3}{3!} + \frac{(7x)^5}{5!} + \cdots\right)$$

$$\hat{f}_T = \begin{pmatrix} 38 \\ 0 \\ c_3 \\ 0 \\ c_5 \\ \vdots \end{pmatrix}$$

$$= 38x - \left(\frac{1}{3!} + \frac{3^3}{3!} + \frac{4 \cdot 7^3}{3!}\right)x^3 + \cdots$$

$c_3$



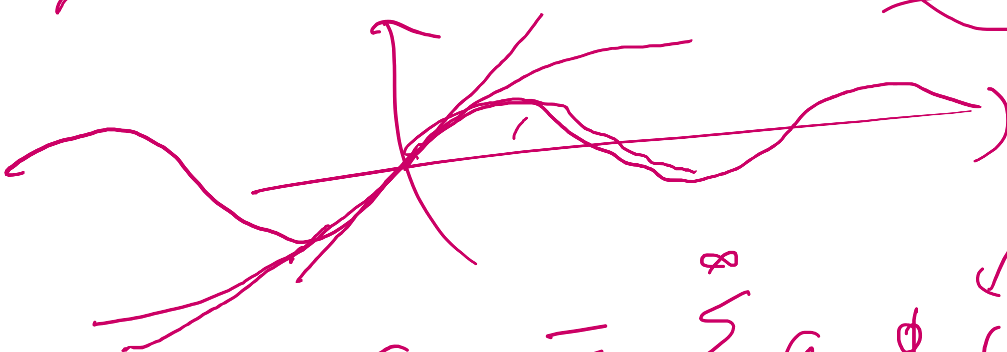

$f = \sin x + 3\sin 3x + 4\sin 7x$


- a vector $v \in E$ is $k$-sparse if $[v]$ has exactly $k$ nonzero values, and $k < \dim(E)$

Exact sparsity vs. approximate

$$f(x) = \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!} + \dots$$

$f_9(x)$ $\quad N=9$




$$\phi_n(x) = x^n$$

$$\sin x = \underbrace{\sum_{n=1}^{\infty} a_n \phi_n(x)}_{f(x)} \approx \underbrace{\sum_{n=1}^{N} a_n \phi_n(x)}_{f_N(x)}$$

$$\| f - f_N(x) \|$$

$$= \| (\sin x - (x - \frac{x^3}{3!}) \quad . \quad . \quad ) \|$$

$$\hat{f} = \begin{pmatrix} 1 \\ 0 \\ 1/6 \\ 0 \\ 1/120 \\ 0/5040 \end{pmatrix} \text{ vs. } \hat{f}_N = \begin{pmatrix} 1 \\ 0 \\ 1/6 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Trunc

$N = 3$

- Truncation impose sparsity.
  (just truncate after a large number.)

- $f(x) = \sin x + 3\sin 3x + 4\sin 12x + 9\sin 47x$

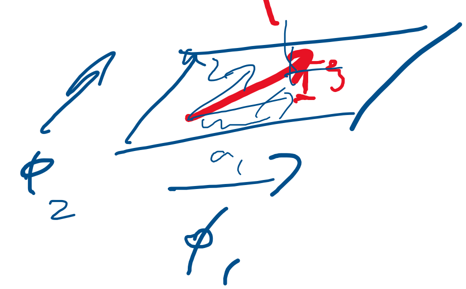
$\hat{f} = \hat{f}_{47} = \begin{pmatrix} 1 \\ 3 \\ 0 \\ \vdots \\ 0 \\ 4 \\ 0 \\ \vdots \\ 0 \\ 9 \\ 0 \\ \vdots \end{pmatrix}$

just 1, 3, 4, 9 and tell you where to put them.

3 Truncation as projection & projection

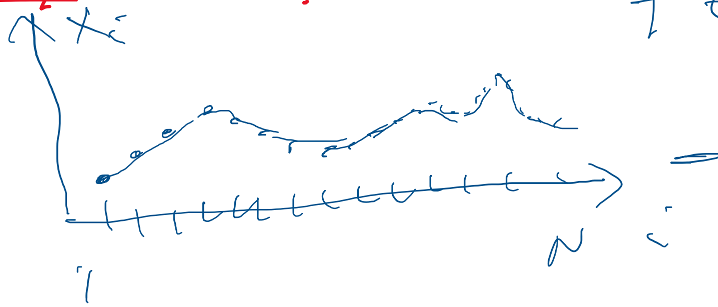
e.g. matrix mult.

$f \in \mathcal{H}$



$\overrightarrow{x} = \begin{pmatrix} 1 \\ 3 \\ 0 \\ 0 \\ 7 \\ 0 \\ 0 \end{pmatrix} \ne \cancel{f(x)}$

$N$

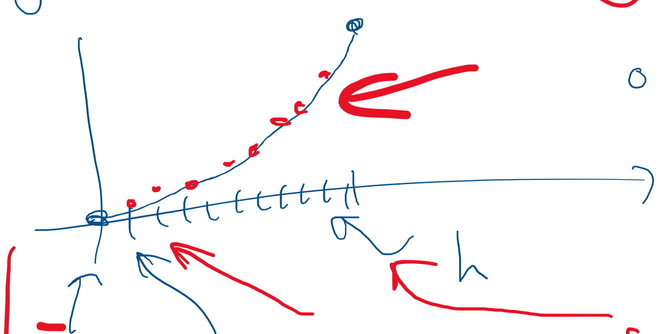


(say from $f(\bar{s}) = \bar{s}^3$)

$x_i = f(s_i)$

$x_i = [x]_i$

$[x]_3 = x_3 = 0$



$0 \le \bar{s} \le 2$

$N = 100$, uniform

$h = \frac{2-0}{N} = \frac{2}{100} = \frac{1}{50}$

$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ (1/50)^3 \\ (2/50)^3 \\ \vdots \end{pmatrix}$

$\bar{s}_1 = 0$ $\quad \bar{s}_2 = \frac{1}{50}$ $\quad \bar{s}_3 = \frac{2}{50}$ $\quad s_{100} = 2$

3.1

$$\underset{n\times 1}{\vec{x}} = \underset{n\times n}{\underline{\Psi}}\ \underset{n\times 1}{\vec{s}} = \underset{n\times n}{\begin{pmatrix} | & | & & | \\ \psi_1 & \psi_2 & \cdots & \psi_n \\ | & | & & | \end{pmatrix}} \underset{n\times 1}{\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}}$$

$n\times 1$ column

$\vec{x} \in \mathrm{col}(\underline{\Psi})$

= linear combo of columns

$$= s_1\psi_1 + s_2\psi_2 + \cdots + s_n\psi_n$$

(linear combo of columns)

$$\vec{x} = \sum_{i=1}^{N} s_i\psi_i \approx \sum_{i=1}^{n} s_i\psi_i, \quad n < N$$

$\underline{\Psi}_n$

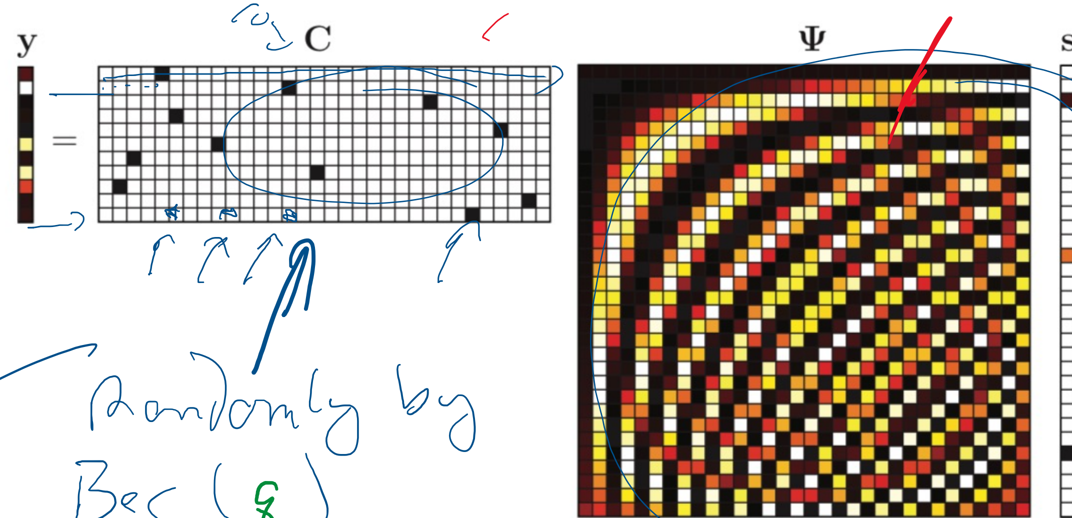
$$\vec{x} = \left(\psi_1\ \psi_2 \cdots \psi_n \,\middle|\, \cancel{\psi_{n+1} \cdots \psi_N}\right) \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ \hline \cancel{s_{n+1}} \\ 0 \end{pmatrix} \rightarrow \underline{\Psi}_N \text{ vs. } \underline{\Psi}_n$$

$$\underline{\Psi}_n = ?\ \underline{\Psi}_N$$

$x = \Psi s$

$\boxed{y = Cx} = \boxed{C\Psi} s$

$\Theta$

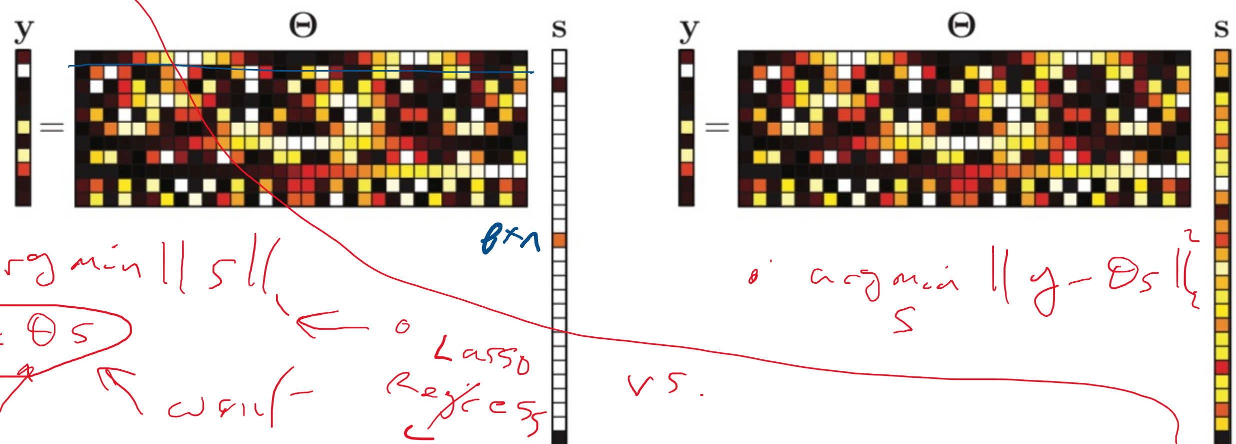


Randomly by Ber($g$), $0 < g < 1$

$= C x$

**Figure 3.4** Schematic of measurements in the compressed sensing framework.

$y = \Theta s$



$p < n$

· $\text{argmin} \|s\|_1$ ← · Lasso Regress

$y = \Theta s$ wait

data

vs.

· $\underset{s}{\text{argmin}} \|y - \Theta s\|_2^2$

Ridge Regress
Tikhonov Regular

(a) Sparse s ($\ell_1$)

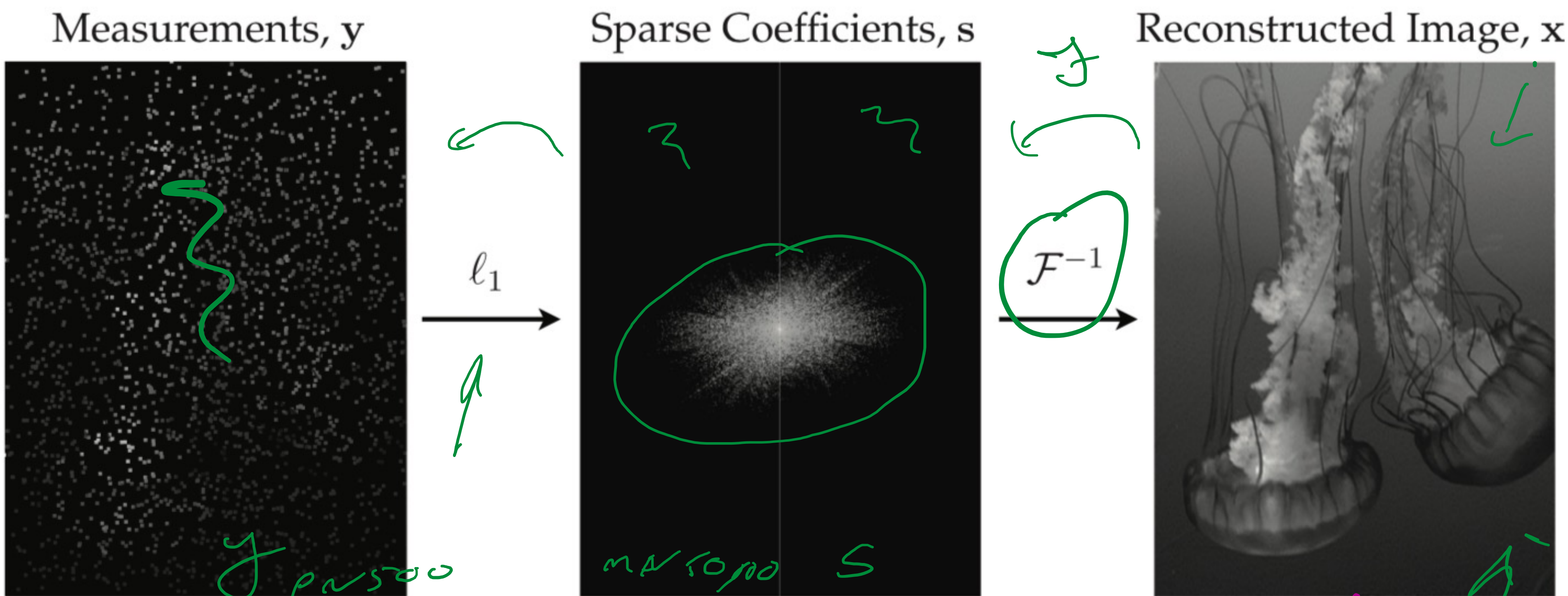
(b) Least-squares s ($\ell_2$)

vs. $J(s) = \|y - \Theta s\|_2^2 + \lambda\|s\|_1$

Regularized

$J(s) = \|y - \Theta s\|_2 + \lambda\|s\|_2^2$

$y = \Theta s$

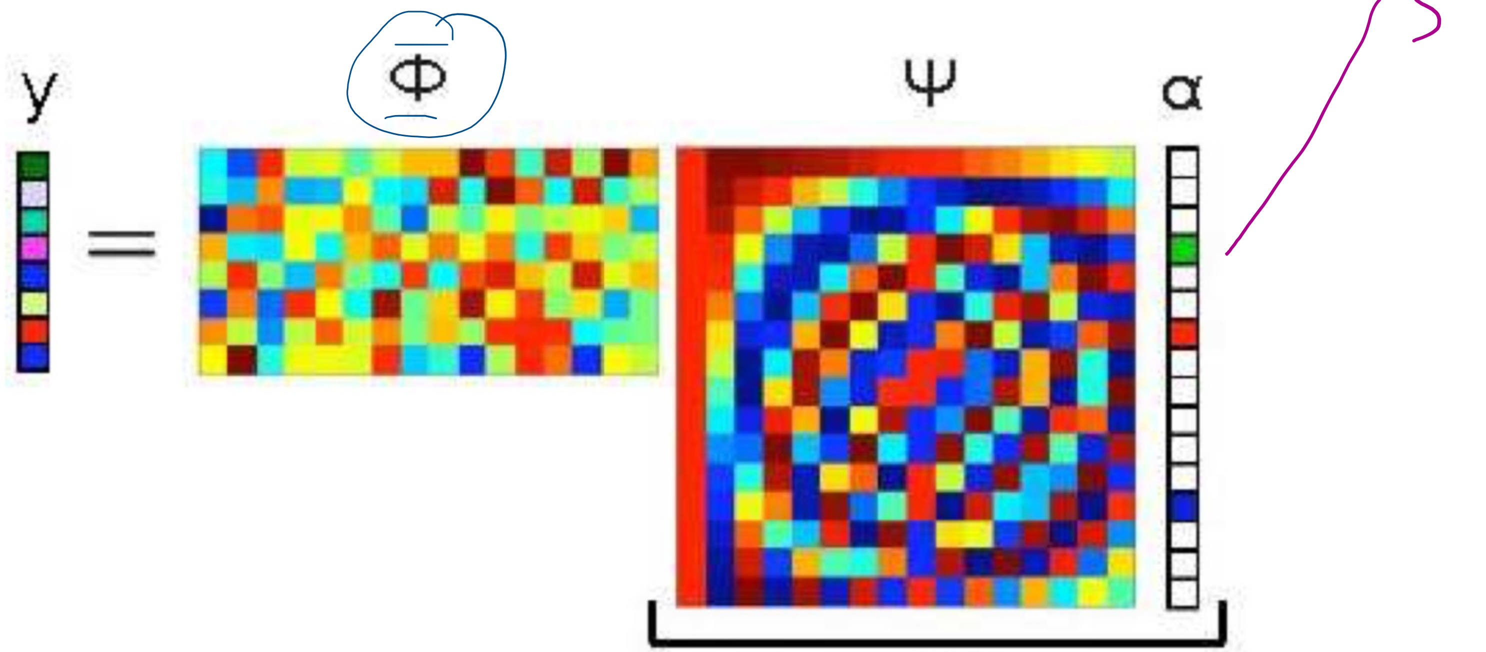



$y = C \Psi s$

$y_{p \times 1} = \Theta s_{n \times 1}$

$x = \Psi s$

Random measurements can be used for signals sparse in any basis.




Signal x in sparse basis Ψ

$\overline{\Theta}$ good enough if "incoherent" to high probability

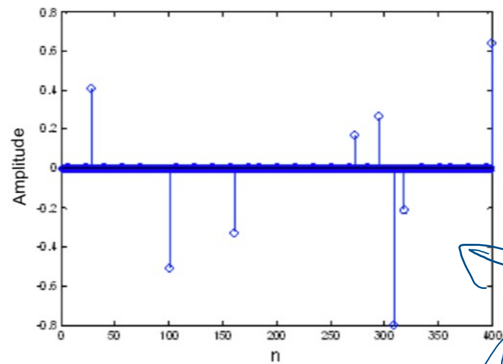
$\overline{\Theta}_{m \times n}$ ~ random (0, 1)

$$y_k = < \phi_k, x >; \quad k = 1, ..., M; \quad \text{with} \quad M \ll N$$

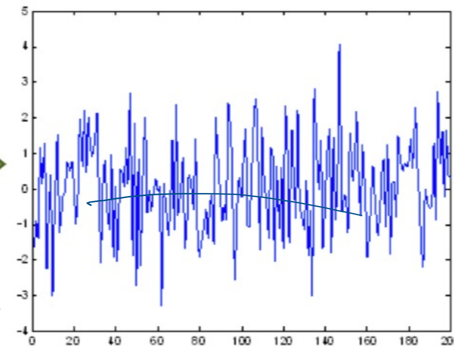
- Need to solve an under determined system of equations $y = \Phi x$.
- Infinitely solutions for the system since $M \ll N$.
- A sparse solution $x$ is recovered from $y$ by solving the following inverse problem

$$(P0): \quad \min_x \|x\|_{\ell_0} \quad s.t. \quad y = \Phi x$$

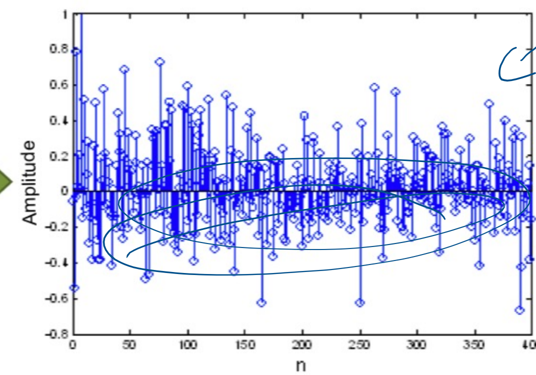
# Example of the recovery of an under determined system of equations:




Original sparse signal



Compressed measurements

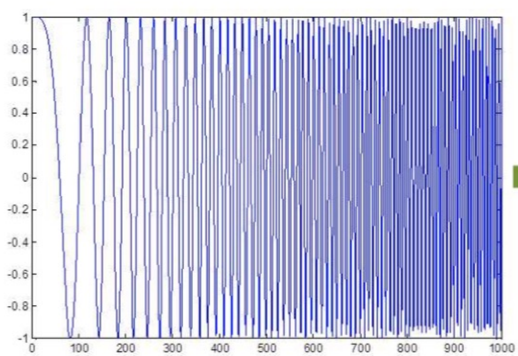



Reconstructed signal using least-squares.
Solution not sparse

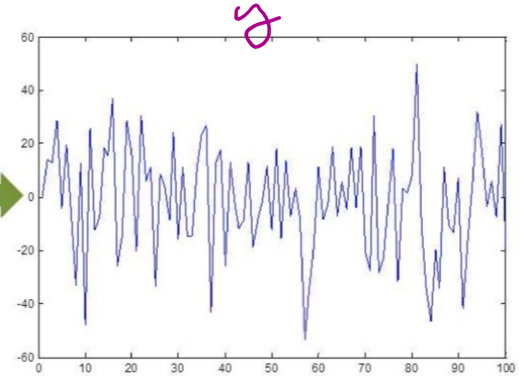
s

y

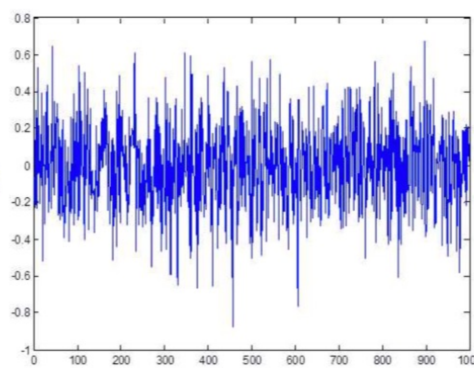
x



Original sparse signal
in Time-Frequency basis



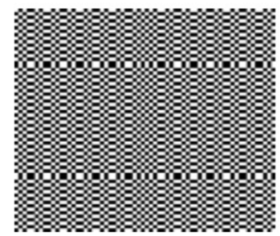
Compressed Measurements



Reconstructed signal using
least-squares

- Sparsity is what makes it possible to recover a signal from undersampled data.
- The number of measurements we need for successful reconstruction depends on the nature of the waveforms $\phi_k$, and $S$

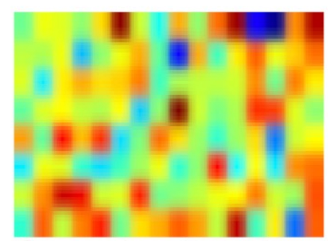
1. Incoherent Orthobasis



Ordered Hadamard Ensemble

Scrambled Block Hadamard Ensemble

2. Random waveforms $\phi_k$



Gaussian Random Ensemble

# Incoherent Orthobasis Example

Example of incoherent basis: the ”spike” basis (identity) and the Fourier basis.

Consider the case where the dictionary is the union of two orthobasis:

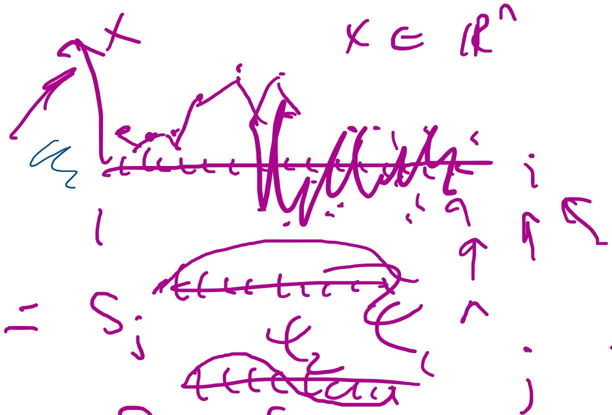
- $I$: the ”spike” basis (identity).
- $F$: the Fourier basis (sinusoids).

$$\Phi = [I; F]$$

where $I$ is a $N \times N$ matrix and $F$ is a $N \times N$ matrix with

$$f_{m,\ell} = \frac{1}{\sqrt{(N)}} e^{j2\pi(m-1)(\ell-1)/N}$$

• Suppose signal $x \in \mathbb{R}^n$



$x_i$; $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

$\xrightarrow{FFT}$ $\langle x, \phi_j \rangle = s_j$

or WT

or other PCA/KL

basis functions $\phi_1, \phi_2, \phi_3$

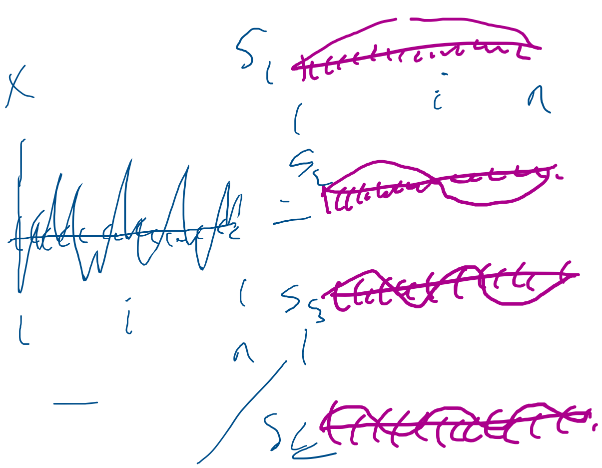
$\vec{x} = s_1\vec{\phi}_1 + s_2\vec{\phi}_2 + \dots + s_k\vec{\phi}_k$

$k = ? = \infty$

$\phi_j(i) = \cancel{\sin j i \pi}$

$i$ is the index of vec.

column vector drawn on its side



$$x = s_1\phi_1 + s_2\phi_2 + s_3\phi_3 + \dots + s_k\phi_k = \begin{pmatrix} | & | & & | \\ \phi_1 & \phi_2 & \cdots & \phi_k \\ | & | & & | \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_k \end{pmatrix}$$

$$\underline{\Phi}_{n\times k}\ \vec{s}_{k\times 1}\ \vec{x}_{n\times 1}$$

$$\boxed{x = \Phi s}$$

$x = \Psi s$ ; $\Psi_{n \times m}$

$s_{m \times 1}$

$x_{m \times 1}$

- choose $k = m$ modes
- a sampler of space for $x$ & $\Psi$:

---

- $$\begin{bmatrix} \\ x \\ \\ \end{bmatrix} = \begin{bmatrix} & & \\ & \Psi & \\ & & \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_k \\ \cancel{s_m} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{not zero } 20{,}000 \\ \text{— } 40{,}000 \end{matrix}$$

Shannon-Nyquist
- choose $k$ at least twice as big as frequency content of $x$

- if $s$ is full (non zeros)
then I need at least to truncate "late"

$$x = \Psi \begin{bmatrix} 0 \\ 0 \\ 0 \\ s_4 \\ 0 \\ 0 \\ 0 \\ s_{75} \\ 0 \\ 0 \\ 0 \\ s_{127} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

40,000 – $m \times 1$

$\vec{s}$ vector

$s_i$ is a scalar

3 is

- k-sparse if $\vec{s}$ has ~~no more~~ exactly. then k nonzero values
- Call sparse if $k \ll m$

$3 \ll 40{,}000$

$\frac{1}{\pi} \sin 4 i \pi$ $\quad$ $\frac{1}{\pi} \sin 75 i \pi$

3

$x$

$$= s_4 \psi_4 + s_{75} \psi_{75} + s_{127} \psi_{127}$$

- sparse basis but likely full in another basis.

$$= \hat{s}_1 \hat{\psi}_1 + \hat{s}_2 \hat{\psi}_2 + \dots \hat{s}_m \hat{\psi}_m$$

- what if $x$ is not <u>sparse</u> w.r.t. B = $\{\psi_i\}_{i=1}^{n}$
  it may be almost sparse w.r.t. B,

$$\vec{x} = s_1\psi_1 + s_2\psi_2 + s_3\psi_3 + \dots + s_m\psi_m$$

- and <u>all</u> or most $s_i \neq 0$ yet

$$s_4 > s_{75} > s_{127} \gg s_i \sim \varepsilon$$

most of the $s_i \sim 0$.

$$s = \begin{pmatrix} 1\cdot 10^{-10} \\ 1\cdot 10^{-8} \\ 3\cdot 10^{-7} \\ 14 \\ 2\cdot 10^{-6} \\ \vdots \\ 3\cdot 10^{-11} \\ 3 \end{pmatrix}$$

- $\boxed{x = \Psi s}$ $s$ coefficients

signal

assume $s$ is $k$-sparse or at least <u>almost $k$-sparse.</u>

- $\|s - \hat{s}\|_2 < \delta$ for $\delta > 0$ small & $\hat{s}$ is sparse

---

Observe $\underset{p \times 1}{y} = \underset{p \times n}{C}\ \underset{n \times 1}{x}$ $p < n$

projection matrix $C$.

signal I want

- $y_{p\times 1} = \underline{\Theta}_{p\times n}\, s_{n\times 1}$ ✓ Solve for $s \in$ | w's $x = \underline{\Psi}\, s$ |

- ? $\cancel{s = \underline{\Theta}^{-1} y}$ $p << n$

singular problem $\Rightarrow$ no unique solution

- inverse problems field say this ill-posed.

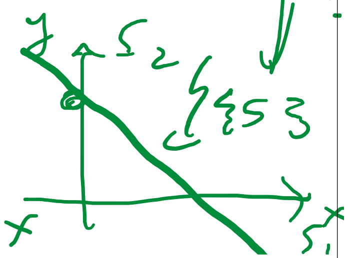
- Hadamard define a well posed problem as one where
  - solution exists
  - uniquely
  - and cts w.r.t. to the data.

else call ill-posed

- Extreme what if $\ell = 1$, $(n = 2)$

Ex: $\underset{1\times 1}{\overset{y}{7}} = \underset{\Theta}{\begin{pmatrix} 1 & 1 \end{pmatrix}}_{1\times 2} \begin{vmatrix} s_1 \\ s_2 \end{vmatrix}$ $\Rightarrow$ $7 = s_1 + s_2$

$s_2 = 7 - s_1$ / $y = 7 - x$




$y = \Theta^T \delta$

But

$7 = \begin{pmatrix} 1 & 1 \end{pmatrix}^2 \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$

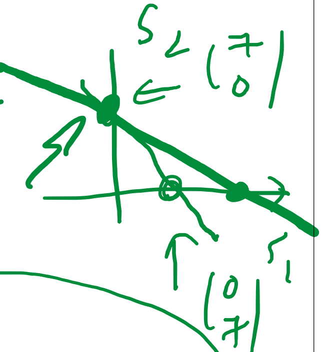
$y = \Theta s$

$p = 1, n = 2$

but true value were $s = \begin{pmatrix} 7 \\ 0 \end{pmatrix}$

if tell you tell me. and $s$ is sparse.

sparse




- let's try. $\vec{s} = \underset{\tilde{s}}{\operatorname{argmin}} \|\tilde{s}\|_0$ subject to $\underline{y = \Theta \tilde{s}}$

$\underline{\left\{ \begin{pmatrix} 7 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 7 \end{pmatrix} \right\}}$

$\|s\|_0 = \#$ of not values.

$\|s\|_1 = |s_1| + |s_2| + \dots + |s_n|$

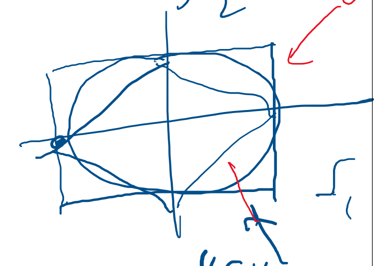
$\|s\|_2 = (s_1^2 + s_2^2 + \dots + s_n^2)^{1/2}$ Euclidean

$\|s\|_\infty = \rightarrow \max_i |s_i|$

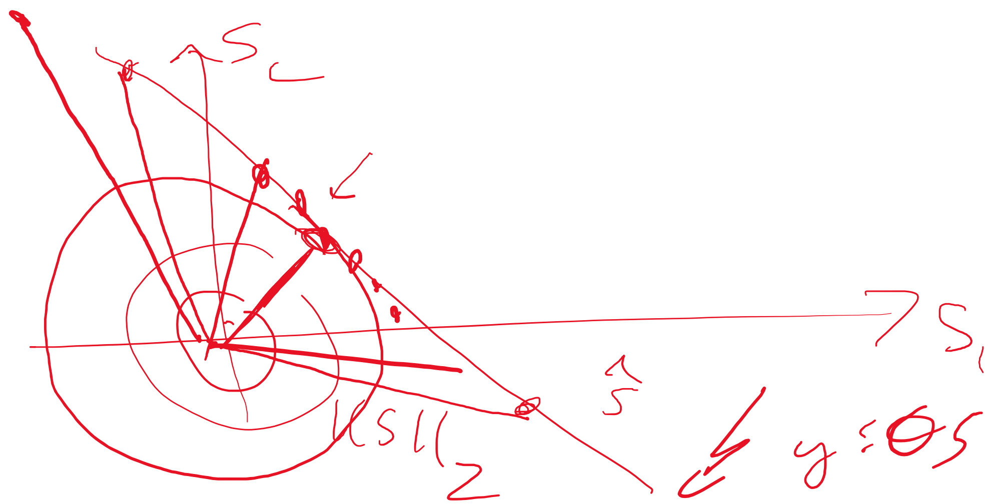
$\|\vec{s}\|_p = (s_1^p + s_2^p + \dots + s_n^p)^{1/p}$

$s \in \mathbb{R}^n$

"Balls" $\|s\|_p = 1$




Want to solve w/ 0-norm — But have to test all vals NP-complete.



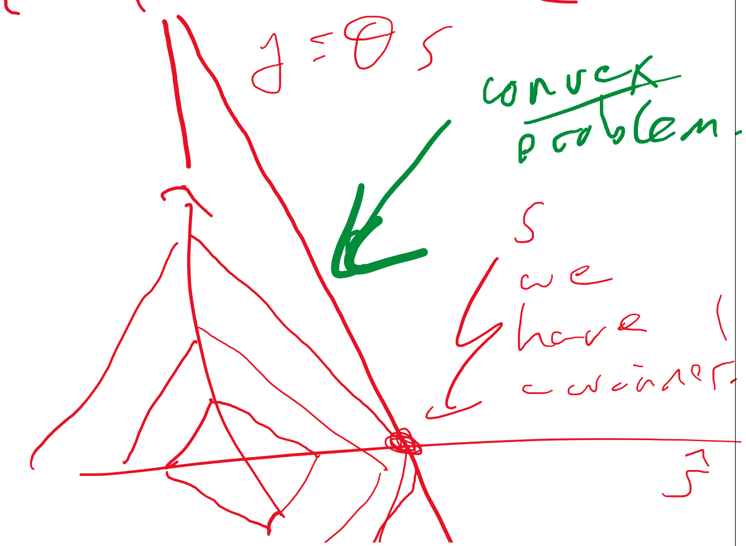

$y = \Theta s$

$p = 1$

$s = \arg\min \|\hat{s}\|_2$ on the circle.
subj
$y = \Theta \hat{s}$

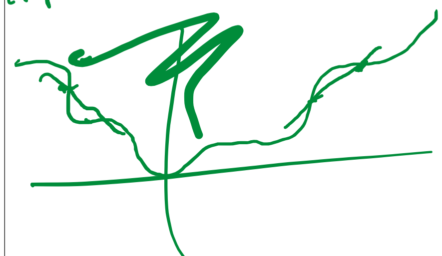
Basis pursuit ... Convex optimization.

$s = \arg\min \|\hat{s}\|_1$
subj
$\rightarrow y = \Theta \hat{s}$



## CS and linprog

not convex



## CVX examples

To solve

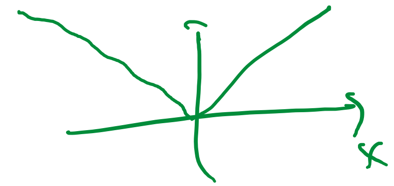
$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

$$\min_x \quad \|x\|_1 \qquad\qquad \min_{x,t} \quad \sum_{i=1}^{n} t_i$$
$$Ax = b \qquad\qquad x \le t$$
$$-x \le t$$
$$Ax = b$$

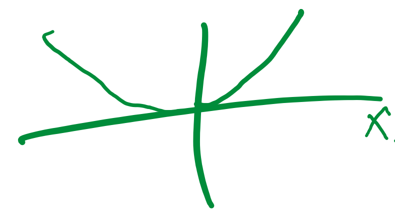
convex problem

Linear Programming

$\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$



$\|x\|_2^2 = x_1^2 + x_2^2 + \dots$



Convex

Use CVX code

```
cvx_begin
  variable x(n);
  variable t(n);
  minimize(sum(t));
  subject to
    x <= t;
    -x <= t;
    A*x == b;
cvx_end
```

- $b = Ax$
- $y = \theta s$

CVX does some standard transformations.

$\cancel{\|x\|_0}$

$\|x\|_1$

To solve

$$\begin{array}{ll} \min & \|x\|_1 \\ \text{subj} & Ax = b \end{array}$$

Use CVX code

```
cvx_begin
  variable x(n);
  minimize(norm(x,1));
  subject to
    A*x == b;
cvx_end
```

$\Rightarrow$

$J(x) = \|x\|_1 + \lambda\|Ax-b\|_2$

$-$ Lasso

$\Rightarrow$

the

$\tilde{J}(x) = \|Ax-b\|_2 + \alpha\|x\|_1$

```matlab
clear all, close all, clc

% Solve y = Theta * s for "s"
n = 1000;  % dimension of s
p = 200;   % number of measurements, dim(y)
Theta = randn(p,n);
y = randn(p,1);

% L1 minimum norm solution s_L1
cvx_begin;
    variable s_L1(n);
    minimize( norm(s_L1,1) );
    subject to
        Theta*s_L1 == y;
cvx_end;

s_L2 = pinv(Theta)*y;  % L2 minimum norm solution s_L2


%%
figure
subplot(3,2,1)
plot(s_L1,'b','LineWidth',1.5)
ylim([-.2 .2]), grid on
subplot(3,2,2)
plot(s_L2,'r','LineWidth',1.5)
ylim([-.2 .2]), grid on
subplot(3,2,[3 5])
[hc,h] = hist(s_L1,[-.1:.01:.1])
bar(h,hc,'b')
axis([-.1 .1 -50 1000])
subplot(3,2,[4 6])
[hc,h] = hist(s_L2,[-.1:.01:.1])
bar(h,hc,'r')
axis([-.1 .1 -20 400])

set(gcf,'Position',[100 100 600 350])
set(gcf,'PaperPositionMode','auto')
%print('-depsc2', '-loose', '../figures/f_chCS_ex03_underdetermined');
```

3.6

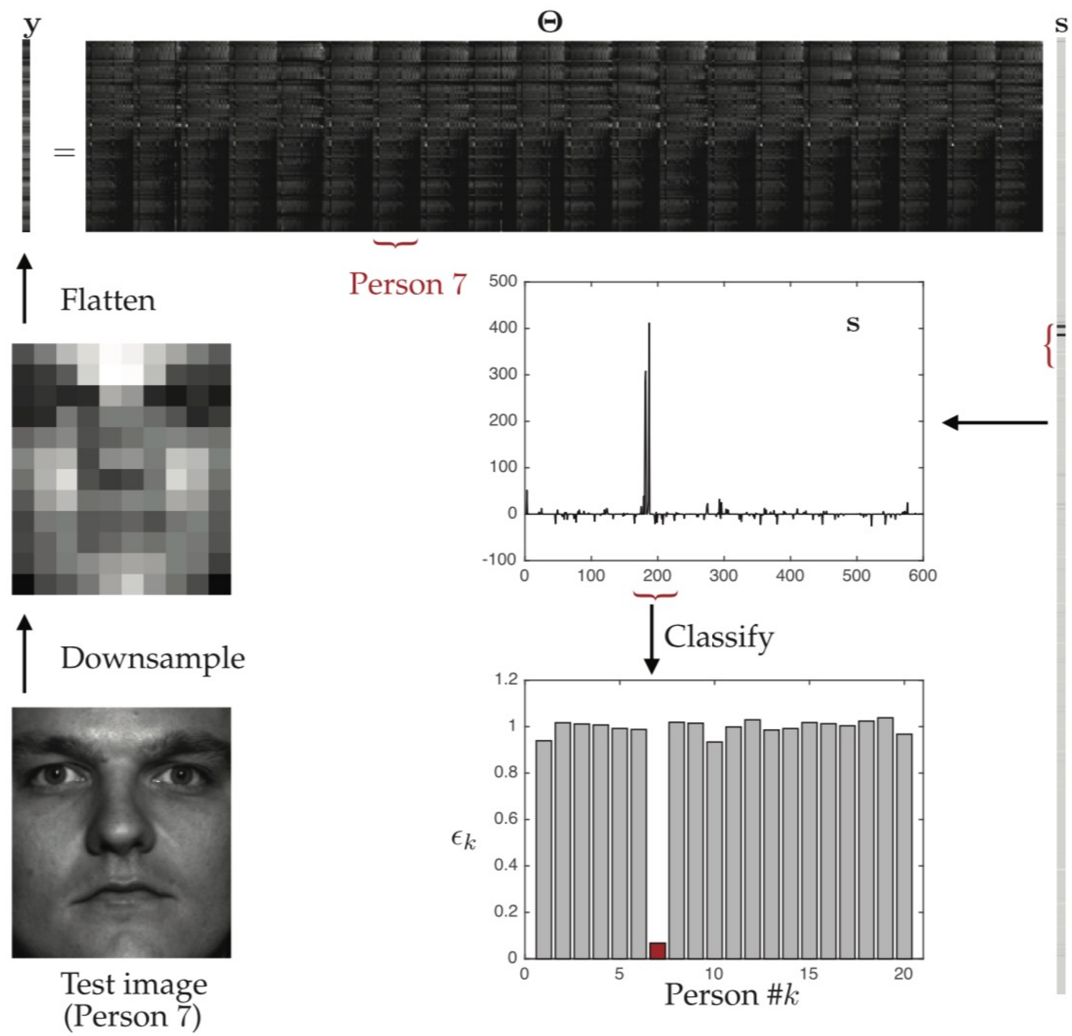



**Figure 3.14** Schematic overview of sparse representation for classification.

/Users/erikbollt/Desktop/Erik/Work/Classes/Clarkson02-/Fall20-Spring21/EE520/CodesAndData/Codes/cvx/CScode.m

```matlab
%Emmanuel Candes, California Institute of Technology, June 6 2007, IMA Summerschool.
%Convex Iteration implementation by Jon Dattorro.
%Failure modes repaired.
clear all, close all
n = 512;                            % Size of signal
m = 64;                             % Number of samples (undersample by a factor 8)

k = 0:n-1;  t = 0:n-1;
F = exp(-i*2*pi*k'*t/n)/sqrt(n);    % Fourier matrix
freq = randsample(n,m);
A = [real(F(freq,:));
     imag(F(freq,:))];              % Incomplete Fourier matrix

S = 28;
support = randsample(n,S);
x0 = zeros(n,1);  x0(support) = randn(S,1);
b = A*x0;

cvx_quiet(true);
%cvx_solver('sedumi');

%convex iteration
y = ones(n,1);
while 1
    % Solve l0 using CVX and Convex Iteration
    cvx_begin
        variable x(n);
        minimize(norm(y.*x,1));
        A*x == b;
    cvx_end

    % update search direction y
    [x_sorted, indices] = sort(abs(x), 'descend');
    y = ones(n,1);
    y(indices(1:S)) = 0;

    cardx = sum(abs(x) > 1e-6)
    if cardx <= S, break, end
end
norm(x - x0)/norm(x0)
```

script    Ln 6    Col 9

/Users/erikbollt/Desktop/Erik/Work/Classes/Clarkson02-/Fall20-Spring21/EE520/CodesAndData/Codes/cvx/CSaslinprog.m

```
% get the vectors
n = 10;
x = rand(n,1);
y = rand(n,1);
% formulate the LP problem in MATLAB
[opt_sol, fval] = linprog([0,ones(1,n)],[x,-eye(n,n); -x, -eye(n,n)], [y;-y]);
c_mat_lp = opt_sol(1);
% for comparison with CVX
% using CVX builtin expression
cvx_begin quiet
    variable c(1)
    minimize(norm(c.*x-y,1))
cvx_end
% LP formulation in CVX
cvx_begin quiet
    variable clp
    variable t(n)
    minimize sum(t)
    subject to
        clp .* x - y <= t
        y-clp.*x <=t
cvx_end
disp (['LP MATLAB solver: obj-val ' num2str(fval) ' c-val ' num2str(c_mat_lp)])
disp (['CVX L1-norm solver: obj-val ' num2str(norm(c.*x-y,1)) ' c-val ' num2str(c)])
disp (['CVX LP formulation: obj-val ' num2str(norm(clp.*x-y,1)) ' c-val ' num2str(clp)])
%end
```