A Numerical Comparison of Compressed Sensing Reconstruction Algorithms

Elaine T. Hale, Wotao Yin, Yin Zhang

Department of Computational and Applied Mathematics
Rice University

McMaster University, ICCOPT II-MOPTA 2007
August 13, 2007

Overview

Overview

Overview

Overview

Classic Sensing and Storage

1. Want n pieces of information.
2. Measure them and store as $x \in \mathbb{R}^n$.
3. Notice that n is large and storage is expensive.
4. Notice that x is *compressible*.
5. Store encoded vector $y \in \mathbb{R}^m$, where $m << n$, instead of x.
6. Decode y to obtain full signal x as needed.

Compressed Sensing

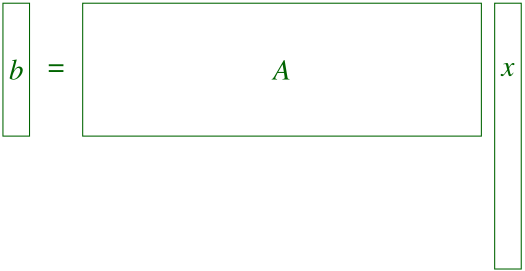
Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.

Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

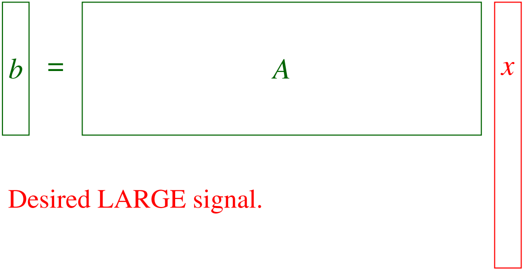
Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.



Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

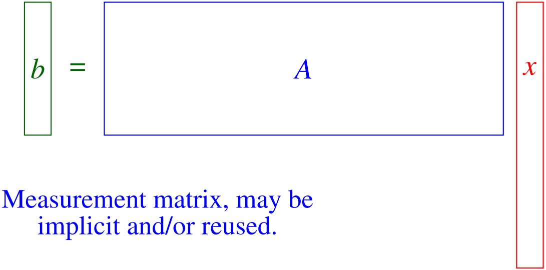
Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.



Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

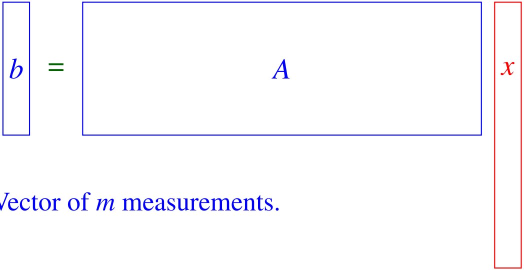
Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.



Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.

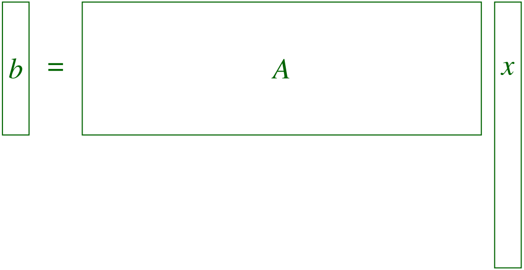


Vector of m measurements.

Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.

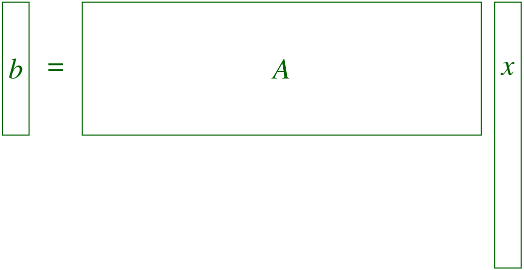


After measurement, only A and b are available. x must be **recovered** in order to use the data.

Compressed Sensing

Can we avoid storing $x \in \mathbb{R}^n$ in the first place?

Yes, if x is *sparse* and we can obtain measurements Ax, where $A \in \mathbb{R}^{m \times n}$, $m << n$, and A is *good*.



If the original signal, z, is not sparse, but $x = \Phi z$ is sparse, take measurements $b = A\Phi z$. Then **recover** x and set $z = \Phi^{-1}x$.

Compressed Sensing Recovery

Recovery is based on the fact that x_s is sparse.

- Greedy Approaches (OMP, StOMP)
  New nonzero component(s) for x^{k+1} correspond to largest component(s) of $|A^T(b - Ax^k)|$.

- Use of ℓ_1 norm in regions where it is equivalent to ℓ_0 “norm".
  In noiseless case solve $\min_x\{\|x\|_1 \,|Ax = b\}$. If x_s is sparse enough, obtain exact recovery and x is also sparsest solution.

A Measurement Model

We will assume

1. A is a given $m \times n$ matrix.
2. x_s is the desired sparse signal (length n, $k << n$ nonzeros).
3. Both x_s and the measurements may be corrupted with Gaussian noise, that is

$$b = A(x_s + \epsilon_1) + \epsilon_2, \tag{1}$$

where ϵ_1 (ϵ_2) is a vector whose elements are *i.i.d.* distributed as $N(0, \sigma_1^2)$ ($N(0, \sigma_2^2)$).

$A, x_s, b, m, n, k, \sigma_1, \sigma_2$

A Measurement Model

We will assume

1. A is a given $m \times n$ matrix.
2. x_s is the desired sparse signal (length n, $k << n$ nonzeros).
3. Both x_s and the measurements may be corrupted with Gaussian noise, that is

$$b = A(x_s + \epsilon_1) + \epsilon_2, \quad (1)$$

where ϵ_1 (ϵ_2) is a vector whose elements are *i.i.d.* distributed as $N(0, \sigma_1^2)$ ($N(0, \sigma_2^2)$).

$$A, x_s, b, m, n, k, \sigma_1, \sigma_2$$

Problem Formulation

Noise? Then $Ax = b$ is not appropriate.

Solution? Minimize or constrain $\|x\|_1$ and $\|Ax - b\|_2^2$.

Our formulation:

$$\min_x \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_M^2, \qquad (2)$$

where M is positive definite.

- If M is inverse covariance matrix of $Ax_s - b$, $\left(\sigma_1^2 AA^T + \sigma_2^2 I\right)^{-1}$, we can derive a recommended value of μ, μ_{rec}.
- If $AA^T = I$ or $\sigma_1 = 0$, this reduces to a standard least-squares formulation.

Problem Formulation

Noise? Then $Ax = b$ is not appropriate.

Solution? Minimize or constrain $\|x\|_1$ and $\|Ax - b\|_2^2$.

Our formulation:

$$\min_x \|x\|_1 + \frac{\mu}{2}\|Ax - b\|_M^2, \tag{2}$$

where M is positive definite.

- If M is inverse covariance matrix of $Ax_s - b$, $\left(\sigma_1^2 AA^T + \sigma_2^2 I\right)^{-1}$, we can derive a recommended value of μ, μ_{rec}.
- If $AA^T = I$ or $\sigma_1 = 0$, this reduces to a standard least-squares formulation.

Fixed Point Continuation

If scale μ, A, and b with $\lambda_{\max}(A^T M A)$, then **fixed point iterations**

$$x^{k+1} = \operatorname{sgn}(x^k - \tau g(x^k)) \circ \max\left\{|x^k - \tau g(x^k)| - \frac{\tau}{\mu}, 0\right\} \qquad (3)$$

converge to a solution of (2) **as long as** $0 < \tau < 2$.

Finite convergence of x^*'s zeros and signs of non-zeros.

Linear convergence of $\{x^k\}$ to x^* as long as $A_N^T A_N$ is full rank, where A_N is a particular $m \times |N|$ sub-matrix of A, or a strict complementarity condition holds.

Fixed Point Continuation

If scale μ, A, and b with $\lambda_{\max}(A^T MA)$, then **fixed point iterations**

$$x^{k+1} = \text{sgn}(x^k - \tau g(x^k)) \circ \max\left\{|x^k - \tau g(x^k)| - \frac{\tau}{\mu}, 0\right\} \quad (3)$$

converge to a solution of (2) **as long as** $0 < \tau < 2$.

Finite convergence of x^*'s zeros and signs of non-zeros.

Linear convergence of $\{x^k\}$ to x^* as long as $A_N^T A_N$ is full rank, where A_N is a particular $m \times |N|$ sub-matrix of A, or a strict complementarity condition holds.

Fixed Point Continuation

If scale μ, A, and b with $\lambda_{\max}(A^T MA)$, then **fixed point iterations**

$$x^{k+1} = \text{sgn}(x^k - \tau g(x^k)) \circ \max\left\{|x^k - \tau g(x^k)| - \frac{\tau}{\mu}, 0\right\} \tag{3}$$

converge to a solution of (2) **as long as** $0 < \tau < 2$.

Finite convergence of x^*'s zeros and signs of non-zeros.

Linear convergence of $\{x^k\}$ to x^* as long as $A_N^T A_N$ is full rank, where A_N is a particular $m \times |N|$ sub-matrix of A, or a strict complementarity condition holds.

Implementation Details

Require: A, b, $\bar{\mu}$; constants τ, f, β, $xtol$, $gtol$

$x_p = 0, \; x = \tau A^T M b, \; \mu = \frac{\tau}{f\|x\|_\infty}$

while $\mu \le \bar{\mu}$ **do**

  while $\frac{\|x - x_p\|_2}{\|x_p\|_2} > xtol\sqrt{\frac{\bar{\mu}}{\mu}}$ **or** $\mu\|g(x_p)\|_\infty - 1 > gtol$ **do**

    $x_p = x$

    $g = A^T M A x - A^T M b$

    $y = x - \tau g$

    $x = \operatorname{sgn}(y) \circ \max\left\{|y| - \frac{\tau}{\mu}, 0\right\}$

  end while

  $\mu = \min\{\beta\mu, \bar{\mu}\}$

end while

Default values: $\tau = \min\left\{-1.665\frac{m}{n} + 2.665, 1.999\right\}$ or $2 - eps$

$f = 0.99, \; \beta = 4, \; xtol = 1\text{E-}4, \; gtol = 0.2$

Implementation Details

Require: A, b, $\bar{\mu}$; constants $\tau, f, \beta, xtol, gtol$

$x_p = 0, \;\; x = \tau A^T M b, \;\; \mu = \frac{\tau}{f\|x\|_\infty}$

while $\mu \leq \bar{\mu}$ **do**

    while $\frac{\|x - x_p\|_2}{\|x_p\|_2} > xtol\sqrt{\frac{\bar{\mu}}{\mu}}$ **or** $\mu\|g(x_p)\|_\infty - 1 > gtol$ **do**

        $x_p = x$

        $g = A^T M A x - A^T M b$

        $y = x - \tau g$

        $x = \operatorname{sgn}(y) \circ \max\left\{|y| - \frac{\tau}{\mu}, 0\right\}$

    end while

    $\mu = \min\{\beta\mu, \bar{\mu}\}$

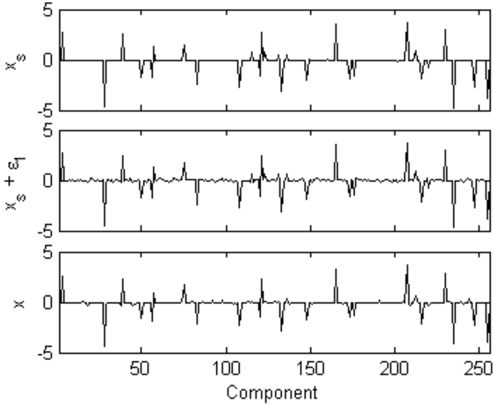
end while

Default values: $\tau = \min\left\{-1.665\frac{m}{n} + 2.665, 1.999\right\}$ or $2 - eps$

$f = 0.99, \;\; \beta = 4, \;\; xtol = 1\text{E-}4, \;\; gtol = 0.2$

Basic Demonstration

- Original, noisy and recovered signals for a compressed sensing problem.
- A is 128×256 partial DCT matrix; $k = 32$; and $\sigma_1 = \sigma_2 = 0.1$.



Continuation and Convergence Rate

- A basic implementation of FPC would run fixed-point iterations at one value of μ: $\bar{\mu}$.
- Presented algorithm solves (2) for an increasing series of μ values: a **continuation** strategy.
- Convergence results are unscathed, and practically the results are much better.

- Run FPC with and without continuation on compressed sensing situation for which we expect exact recovery:
  - Gaussian A matrix
  - $\sigma_1 = \sigma_2 = 0$, $n = 2048$
  - Number of measurements is $m = 0.4n$
  - Number of non-zeros in x_s is $k = 0.05m$
  - $\bar{\mu}$ =1E3 and 1E9.

Plot $|\text{supp}(x^k)|$ and $\log \|x - x_s\|/\|x_s\|$ v. iteration count.

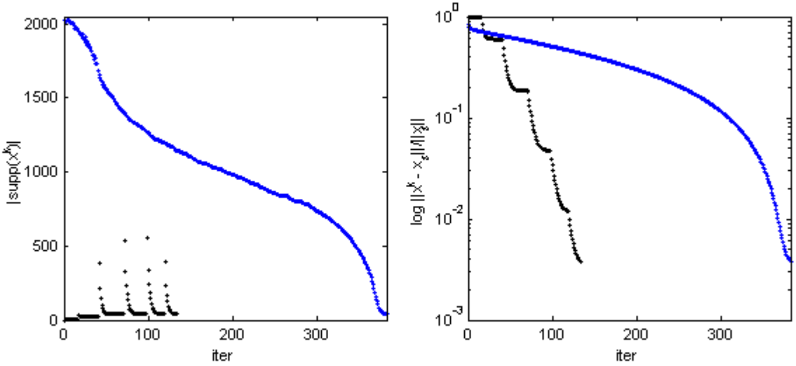
Continuation and Convergence Rate

- A basic implementation of FPC would run fixed-point iterations at one value of μ: $\bar{\mu}$.
- Presented algorithm solves (2) for an increasing series of μ values: a **continuation** strategy.
- Convergence results are unscathed, and practically the results are much better.

- Run FPC with and without continuation on compressed sensing situation for which we expect exact recovery:
  - Gaussian A matrix
  - $\sigma_1 = \sigma_2 = 0$, $n = 2048$
  - Number of measurements is $m = 0.4n$
  - Number of non-zeros in x_s is $k = 0.05m$
  - $\bar{\mu}$ =1E3 and 1E9.

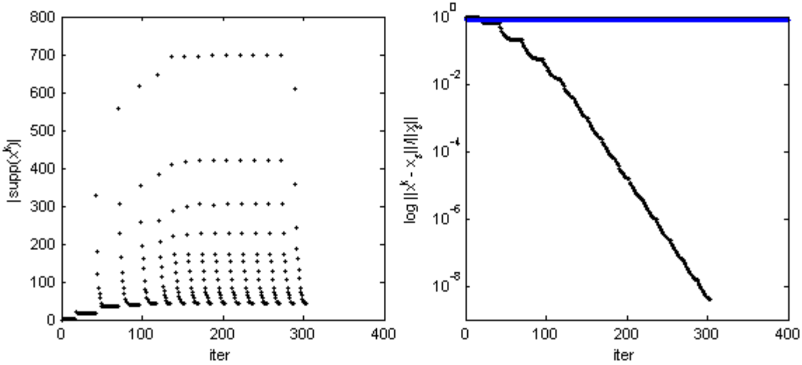
Plot $|\text{supp}(x^k)|$ and $\log \|x - x_s\|/\|x_s\|$ v. iteration count.

Continuation & Convergence Plots



$\bar{\mu}$ =1E3. With and without continuation.

Continuation & Convergence Plots



$\bar{\mu}$ =1E9. With and without continuation.

Results are truncated with respect to iteration number and $|\mathrm{supp}(x^k)|$. Without continuation, $|\mathrm{supp}(x^k)| = 2048$.

Numerical Results, Preliminaries

We compared FPC with and without line search to three algorithms:

- **StOMP** (Donoho et al., 2006). Greedy approach that uses statistical arguments to determine the significant components of $|A^T(b - Ax^k)|$. Updates nonzero components by solving reduced least-squares problem.
- **l1_ls** (Kim et al., 2007). Interior-point method for (2) that uses a preconditioned conjugate gradient (PCG) method to approximately solve linear systems in a truncated-Newton framework.
- **GPSR** (Figueiredo et al., 2007). Gradient projection method applied to reformulation of (2) as a bound-constrained quadratic program. There are several variants. Here we compare to Monotone GPSR-BB.

Numerical Results, Preliminaries, cont.

Two types of A matrices:

- **Gaussian**. Explicit matrix. Matrix-vector multiplications cost $O(mn)$. $AA^T \neq I$, but analytic bounds for $\lambda_{\max}(AA^T)$ and $\lambda_{\min}(AA^T)$ exist.
- **Partial DCT**. Implicit matrix. Matrix-vector multiplications are implemented with a fast transform and cost $O(n \log n)$. $AA^T = I$.

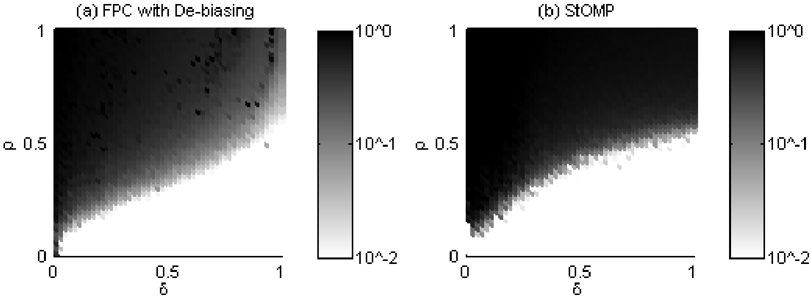
Varying noise levels σ_1 (on the original signal) and σ_2 (on the measurements).

Varying measurement ($\delta = m/n$) and sparsity ($\rho = k/m$) levels.

Phase Plots

- Phase plots depict speed or accuracy of compressed sensing recovery as a function of $\delta = m/n$ and $\rho = k/m$.
- Here, $n = 512$, each point is average over 30 runs, and $\delta = 0.02 : 0.02 : 1$, $\rho = 0.02 : 0.02 : 1$.
- Show plots for StOMP and FPC. Plots for l1_ls and GPSR should be similar to FPC's.
- StOMP's Gaussian A matrices' columns were scaled to unit norm. (Necessary for their statistical thresholding.)

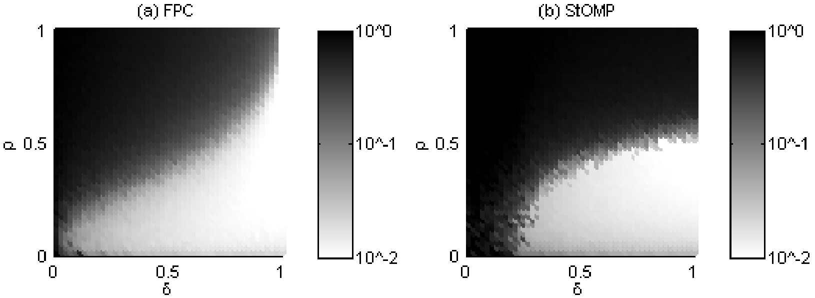
Phase Plots for Gaussian A, $\sigma_1 = 0$, $\sigma_2 = 1\text{E-}8$



Shading indicates value of $\frac{\|x-x_s\|}{\|x_s\|}$. All values larger than 1 are black and all less than 1E-2 are white.

$\delta = m/n$ (measurement ratio), $\rho = k/m$ (sparsity ratio).

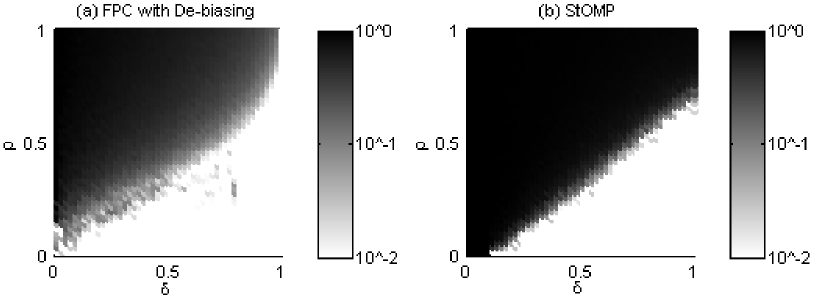
Phase Plots for Gaussian A, $\sigma_1 = 1\text{E-}2$, $\sigma_2 = 1\text{E-}8$



Shading indicates value of $\frac{\|x - x_s\|}{\|x_s\|}$. All values larger than 1 are black and all less than 1E-2 are white.

$\delta = m/n$ (measurement ratio), $\rho = k/m$ (sparsity ratio).

Phase Plots for DCT A, $\sigma_1 = 0$, σ_2 = 1E-8



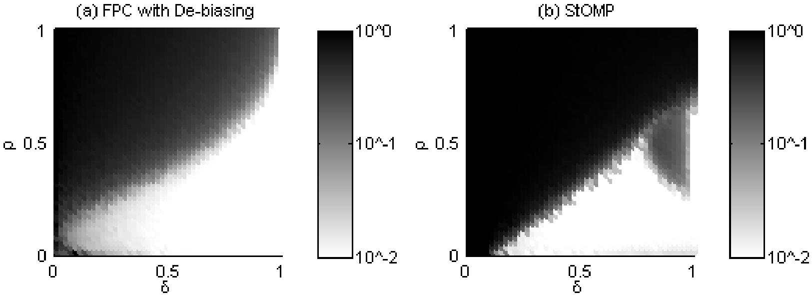
Shading indicates value of $\frac{\|x - x_s\|}{\|x_s\|}$. All values larger than 1 are black and all less than 1E-2 are white.

$\delta = m/n$ (measurement ratio), $\rho = k/m$ (sparsity ratio).

Phase Plots for DCT A, $\sigma_1 = $ 1E-2, $\sigma_2 = $ 1E-8

A Numerical Comparison of Compressed Sensing Reconstruction Algorithms

Elaine T. Hale



Shading indicates value of $\frac{\|x - x_s\|}{\|x_s\|}$. All values larger than 1 are black and all less than 1E-2 are white.

$\delta = m/n$ (measurement ratio), $\rho = k/m$ (sparsity ratio).

Remarks on Phase Plots

- Phase transition curve depends on matrix type, noise levels and recovery algorithm.
- When there is little noise, superiority of FPC or StOMP may depend on (δ, ρ).
- FPC is generally superior at higher noise levels, especially for extreme m (large or small).
- However, StOMP is accurate in the face of noise for many (δ, ρ) combinations, as claimed in Donoho et al. (2006).

μ Study

- Examined accuracy and speed of FPC, GPSR and l1_ls as a function of μ.
- Results for l1_ls are not shown. In general, l1_ls is just as accurate as FPC, but much slower.
- Nonzero components of x_s chosen uniformly at random. Nonzero entries from $N(0,4)$.
- Each data point is the average of 10 runs.
- GPSR was run to value of objective function found by FPC. GPSR's default stopping criterion results in premature exit when μ is large.

Line Search

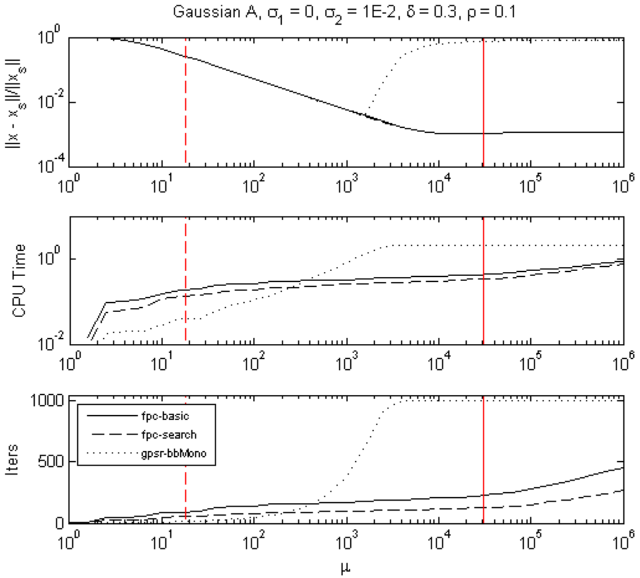
Given previous signal estimate, x_p, and fixed-point iteration result, x, one-dimensional line search between them is:

$$\min_{\alpha} ||\hat{x}||_1 + \frac{\mu}{2}||A\hat{x} - b||_M^2 \quad \text{s.t. } \hat{x} = x_p + \alpha(x - x_p) \tag{4}$$

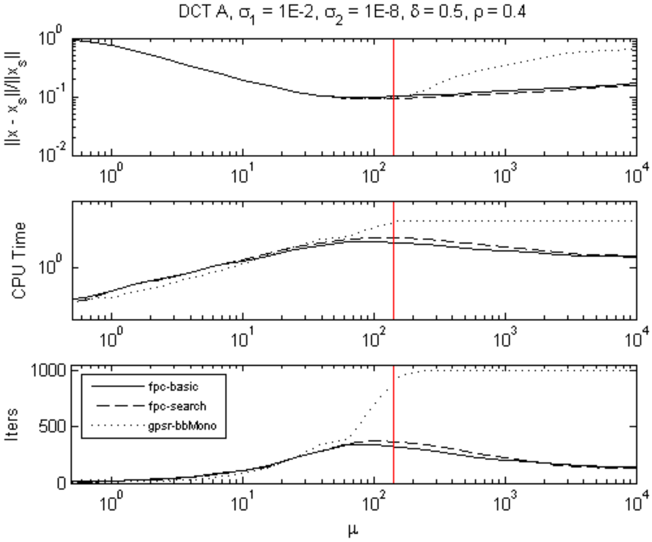
This problem can be solved analytically without any additional matrix-vector multiplications.

It is often advantageous to solve for α^* and implement $x_p + \alpha^*(x - x_p)$.

An Easy Case: Sparse and No Signal Noise



A Hard Case: Less Sparse and Signal Noise



Remarks on Accuracy and Speed as a Function of μ

- Recommended M and μ work well. Order of magnitude estimates for σ_1 and σ_2 are sufficient.
- For large enough μ, GPSR convergence slows down considerably, and GPSR stops recovering x_s to high accuracy because it reaches its maximum iterations.
- GPSR is faster than FPC for small μ, but a cross-over occurs, generally before μ_{rec}.
- Authors of GPSR are currently implementing a continuation strategy–this may fix their problems at large μ.
- In the meantime, GPSR is competitive with FPC IF $\mu < \mu_{\text{rec}}$ and *de-biasing* is used.

De-biasing

To *de-bias* the FPC results:

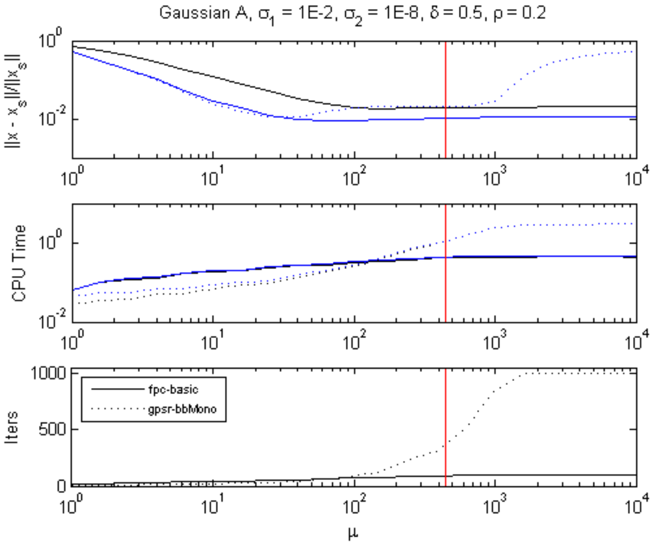
1. Identify $N = \{i | x_i^* \neq 0\}$.
2. If $0 < |N| \leq m$, solve the overdetermined least squares problem
$$y^* = \arg\min_y \|A_N y - b\|_2^2.$$
3. Set $x_N = y^*$ and the rest of x's components to zero.

Easy to find N when μ is small ... for large μ results can deteriorate if "0" not defined properly. A guideline:

$$N = \left\{ i \,\middle|\, |x_i^*| > 3\sqrt{\sigma_1^2 + \frac{\sigma_2^2}{\lambda_{\min}(AA^T)}} \right\},$$

which depends on good estimates of σ_1 and σ_2.

Effects of Debiasing (in Blue)



Timing Study

- Like μ Study except solve a series of problems with increasing n rather than one problem with increasing $\bar{\mu}$.
- Compare to StOMP, in addition to GPSR and l1_ls.
- For the StOMP runs with Gaussian A, the columns of A were scaled to unit norm.

Gaussian A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 8192$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		4.1E-7 $n^{2.06}$, 52.6	2.6E-6 $n^{1.93}$, 118
	fpc3-search		1.8E-6 $n^{1.81}$, 30.8	7.8E-6 $n^{1.75}$, 80.0
	gpsr-bbMono		7.2E-7 $n^{1.88}$, 20.0	not rec.
	stomp		not rec.	1.4E-9 $n^{2.95}$, 524
$\rho = 0.2$	fpc3-basic	4.4E-7 $n^{1.93}$, 16.8	2.0E-6 $n^{1.84}$, 37.3	2.0E-6 $n^{1.86}$, 47.9
	fpc3-search	7.0E-6 $n^{1.53}$, 8.01	1.4E-5 $n^{1.54}$, 20.1	1.1E-6 $n^{1.88}$, 31.7
	gpsr-bbMono	3.9E-8 $n^{2.05}$, 3.78	2.6E-7 $n^{2.13}$, 69.0	4.1E-7 $n^{2.14}$, 125
	stomp	not rec.	4.8E-9 $n^{2.71}$, 198	6.7E-9 $n^{2.68}$, 396
$\rho = 0.1$	fpc3-basic	1.3E-6 $n^{1.77}$, 10.6	9.0E-7 $n^{1.83}$, 13.5	1.1E-6 $n^{1.84}$, 19.7
	fpc3-search	9.4E-6 $n^{1.45}$, 4.98	1.6E-6 $n^{1.70}$, 8.52	1.4E-5 $n^{1.49}$, 14.6
	gpsr-bbMono	9.1E-8 $n^{2.05}$, 11.2	4.0E-8 $n^{2.25}$, 26.4	9.7E-8 $n^{2.22}$, 52.4
	stomp	not rec.	5.6E-10 $n^{2.92}$, 145	4.9E-9 $n^{2.64}$, 146

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 8192.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/100$ or $\mu_{\text{rec}}/1000$.

Gaussian A, $\sigma_1 = 0$, $\sigma_2 =$ 1E-2, $n_{\max} = 8192$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		4.1E-7 $n^{2.06}$, 52.6	2.6E-6 $n^{1.93}$, 118
	fpc3-search		1.8E-6 $n^{1.81}$, 30.8	7.8E-6 $n^{1.75}$, 80.0
	gpsr-bbMono		7.2E-7 $n^{1.88}$, 20.0	not rec.
	stomp		not rec.	1.4E-9 $n^{2.95}$, 524
$\rho = 0.2$	fpc3-basic	4.4E-7 $n^{1.93}$, 16.8	2.0E-6 $n^{1.84}$, 37.3	2.0E-6 $n^{1.86}$, 47.9
	fpc3-search	7.0E-6 $n^{1.53}$, 8.01	1.4E-5 $n^{1.54}$, 20.1	1.1E-6 $n^{1.88}$, 31.7
	gpsr-bbMono	3.9E-8 $n^{2.05}$, 3.78	2.6E-7 $n^{2.13}$, 69.0	4.1E-7 $n^{2.14}$, 125
	stomp	not rec.	4.8E-9 $n^{2.71}$, 198	6.7E-9 $n^{2.68}$, 396
$\rho = 0.1$	fpc3-basic	1.3E-6 $n^{1.77}$, 10.6	9.0E-7 $n^{1.83}$, 13.5	1.1E-6 $n^{1.84}$, 19.7
	fpc3-search	9.4E-6 $n^{1.45}$, 4.98	1.6E-6 $n^{1.70}$, 8.52	1.4E-5 $n^{1.49}$, 14.6
	gpsr-bbMono	9.1E-8 $n^{2.05}$, 11.2	4.0E-8 $n^{2.25}$, 26.4	9.7E-8 $n^{2.22}$, 52.4
	stomp	not rec.	5.6E-10 $n^{2.92}$, 145	4.9E-9 $n^{2.64}$, 146

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 8192.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/100$ or $\mu_{\text{rec}}/1000$.

Gaussian A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 8192$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		4.1E-7 $n^{2.06}$, 52.6	2.6E-6 $n^{1.93}$, 118
	fpc3-search		1.8E-6 $n^{1.81}$, 30.8	7.8E-6 $n^{1.75}$, 80.0
	gpsr-bbMono		7.2E-7 $n^{1.88}$, 20.0	not rec.
	stomp		not rec.	1.4E-9 $n^{2.95}$, 524
$\rho = 0.2$	fpc3-basic	4.4E-7 $n^{1.93}$, 16.8	2.0E-6 $n^{1.84}$, 37.3	2.0E-6 $n^{1.86}$, 47.9
	fpc3-search	7.0E-6 $n^{1.53}$, 8.01	1.4E-5 $n^{1.54}$, 20.1	1.1E-6 $n^{1.88}$, 31.7
	gpsr-bbMono	3.9E-8 $n^{2.05}$, 3.78	2.6E-7 $n^{2.13}$, 69.0	4.1E-7 $n^{2.14}$, 125
	stomp	not rec.	4.8E-9 $n^{2.71}$, 198	6.7E-9 $n^{2.68}$, 396
$\rho = 0.1$	fpc3-basic	1.3E-6 $n^{1.77}$, 10.6	9.0E-7 $n^{1.83}$, 13.5	1.1E-6 $n^{1.84}$, 19.7
	fpc3-search	9.4E-6 $n^{1.45}$, 4.98	1.6E-6 $n^{1.70}$, 8.52	1.4E-5 $n^{1.49}$, 14.6
	gpsr-bbMono	9.1E-8 $n^{2.05}$, 11.2	4.0E-8 $n^{2.25}$, 26.4	9.7E-8 $n^{2.22}$, 52.4
	stomp	not rec.	5.6E-10 $n^{2.92}$, 145	4.9E-9 $n^{2.64}$, 146

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 8192.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/100$ or $\mu_{\text{rec}}/1000$.

DCT A, $\sigma_1 = 0$, $\sigma_2 = 1E\text{-}2$, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		2.4E-4 $n^{1.03}$, 364	7.7E-5 $n^{1.05}$, 171
	fpc3-search		1.5E-4 $n^{1.05}$, 309	1.1E-4 $n^{1.02}$, 178
	gpsr-bbMono		1.9E-4 $n^{1.03}$, 298	7.1E-5 $n^{1.03}$, 119
	stomp		not rec.	1.8E-4 $n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	3.1E-4 $n^{1.05}$, $\gg$ 300	1.0E-4 $n^{1.04}$, 178	5.9E-5 $n^{1.03}$, 107
	fpc3-search	1.8E-4 $n^{1.05}$, 383	7.0E-5 $n^{1.06}$, 169	4.6E-5 $n^{1.06}$, 112
	gpsr-bbMono	1.2E-4 $n^{1.04}$, 206	7.4E-5 $n^{1.02}$, 105	7.1E-5 $n^{0.98}$, 61.0
	stomp	not rec.	1.7E-4 $n^{1.08}$, 519	5.8E-5 $n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	1.8E-4 $n^{1.03}$, 302	5.3E-5 $n^{1.05}$, 119	3.3E-5 $n^{1.05}$, 71.6
	fpc3-search	5.7E-5 $n^{1.09}$, 206	3.6E-5 $n^{1.07}$, 103	3.9E-5 $n^{1.04}$, 72.6
	gpsr-bbMono	3.9E-5 $n^{1.04}$, 69.8	2.9E-5 $n^{1.04}$, 51.6	2.3E-5 $n^{1.05}$, 46.8
	stomp	not rec.	6.4E-5 $n^{1.08}$, 199	4.3E-5 $n^{1.12}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/10$.

DCT A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		2.4E-4 $n^{1.03}$, 364	7.7E-5 $n^{1.05}$, 171
	fpc3-search		1.5E-4 $n^{1.05}$, 309	1.1E-4 $n^{1.02}$, 178
	gpsr-bbMono		1.9E-4 $n^{1.03}$, 298	7.1E-5 $n^{1.03}$, 119
	stomp		not rec.	1.8E-4 $n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	3.1E-4 $n^{1.05}$, $\gg$ 300	1.0E-4 $n^{1.04}$, 178	5.9E-5 $n^{1.03}$, 107
	fpc3-search	1.8E-4 $n^{1.05}$, 383	7.0E-5 $n^{1.06}$, 169	4.6E-5 $n^{1.06}$, 112
	gpsr-bbMono	1.2E-4 $n^{1.04}$, 206	7.4E-5 $n^{1.02}$, 105	7.1E-5 $n^{0.98}$, 61.0
	stomp	not rec.	1.7E-4 $n^{1.08}$, 519	5.8E-5 $n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	1.8E-4 $n^{1.03}$, 302	5.3E-5 $n^{1.05}$, 119	3.3E-5 $n^{1.05}$, 71.6
	fpc3-search	5.7E-5 $n^{1.09}$, 206	3.6E-5 $n^{1.07}$, 103	3.9E-5 $n^{1.04}$, 72.6
	gpsr-bbMono	3.9E-5 $n^{1.04}$, 69.8	2.9E-5 $n^{1.04}$, 51.6	2.3E-5 $n^{1.05}$, 46.8
	stomp	not rec.	6.4E-5 $n^{1.08}$, 199	4.3E-5 $n^{1.12}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/10$.

DCT A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		2.4E-4 $n^{1.03}$, 364	7.7E-5 $n^{1.05}$, 171
	fpc3-search		1.5E-4 $n^{1.05}$, 309	1.1E-4 $n^{1.02}$, 178
	gpsr-bbMono		1.9E-4 $n^{1.03}$, 298	7.1E-5 $n^{1.03}$, 119
	stomp		not rec.	1.8E-4 $n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	3.1E-4 $n^{1.05}$, $\gg$ 300	1.0E-4 $n^{1.04}$, 178	5.9E-5 $n^{1.03}$, 107
	fpc3-search	1.8E-4 $n^{1.05}$, 383	7.0E-5 $n^{1.06}$, 169	4.6E-5 $n^{1.06}$, 112
	gpsr-bbMono	1.2E-4 $n^{1.04}$, 206	7.4E-5 $n^{1.02}$, 105	7.1E-5 $n^{0.98}$, 61.0
	stomp	not rec.	1.7E-4 $n^{1.08}$, 519	5.8E-5 $n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	1.8E-4 $n^{1.03}$, 302	5.3E-5 $n^{1.05}$, 119	3.3E-5 $n^{1.05}$, 71.6
	fpc3-search	5.7E-5 $n^{1.09}$, 206	3.6E-5 $n^{1.07}$, 103	3.9E-5 $n^{1.04}$, 72.6
	gpsr-bbMono	3.9E-5 $n^{1.04}$, 69.8	2.9E-5 $n^{1.04}$, 51.6	2.3E-5 $n^{1.05}$, 46.8
	stomp	not rec.	6.4E-5 $n^{1.08}$, 199	4.3E-5 $n^{1.12}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576.

FPC and GPSR include time for de-biasing, and were run at $\mu = \mu_{\text{rec}}/10$.

DCT A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		0.00053 $n^{1.02}$, 367[a]	6.4E-5 $n^{1.05}$, 146
	fpc3-search		0.00061 $n^{1.02}$, 395[a]	6.9E-5 $n^{1.05}$, 154
	gpsr-bbMono		0.0015 $n^{1.03}$, $\gg$ 300[a]	6.3E-5 $n^{1.03}$, 103
	stomp		not rec.	1.8E-4 $n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	6.3E-4 $n^{1.04}$, 555[a]	1.9E-4 $n^{1.03}$, 294	4.8E-5 $n^{1.04}$, 89.6
	fpc3-search	4.0E-4 $n^{1.02}$, 293[a]	1.6E-4 $n^{1.03}$, 267	3.7E-5 $n^{1.06}$, 94.6
	gpsr-bbMono	1.3E-3 $n^{1.04}$, $\gg$ 300[a]	6.6E-4 $n^{1.01}$, $\gg$ 300	3.1E-5 $n^{1.03}$, 47.3
	stomp	not rec.	1.7E-4 $n^{1.08}$, 519	5.8E-5 $n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	2.5E-4 $n^{1.04}$, 481	8.2E-5 $n^{1.04}$, 159	5.9E-5 $n^{1.03}$, 95.8
	fpc3-search	1.6E-4 $n^{1.04}$, 321	8.0E-5 $n^{1.05}$, 158	6.1E-5 $n^{1.03}$, 95.9
	gpsr-bbMono	4.1E-4 $n^{1.04}$, $\gg$ 300	2.2E-4 $n^{1.03}$, 356	8.4E-5 $n^{1.03}$, 131
	stomp	not rec.	6.4E-5 $n^{1.08}$, 199	2.9E-5 $n^{1.15}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576, except for [a]. Then n = 524,288.

FPC and GPSR were run at $\mu = \mu_{\text{rec}}$, and their results were not de-biased.

DCT A, $\sigma_1 = 0$, $\sigma_2 = 1\text{E-}2$, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		$0.00053\ n^{1.02}$, 367[a]	$6.4\text{E-}5\ n^{1.05}$, 146
	fpc3-search		$0.00061\ n^{1.02}$, 395[a]	$6.9\text{E-}5\ n^{1.05}$, 154
	gpsr-bbMono		$0.0015\ n^{1.03}$, $\gg$ 300[a]	$6.3\text{E-}5\ n^{1.03}$, 103
	stomp		not rec.	$1.8\text{E-}4\ n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	$6.3\text{E-}4\ n^{1.04}$, 555[a]	$1.9\text{E-}4\ n^{1.03}$, 294	$4.8\text{E-}5\ n^{1.04}$, 89.6
	fpc3-search	$4.0\text{E-}4\ n^{1.02}$, 293[a]	$1.6\text{E-}4\ n^{1.03}$, 267	$3.7\text{E-}5\ n^{1.06}$, 94.6
	gpsr-bbMono	$1.3\text{E-}3\ n^{1.04}$, $\gg$ 300[a]	$6.6\text{E-}4\ n^{1.01}$, $\gg$ 300	$3.1\text{E-}5\ n^{1.03}$, 47.3
	stomp	not rec.	$1.7\text{E-}4\ n^{1.08}$, 519	$5.8\text{E-}5\ n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	$2.5\text{E-}4\ n^{1.04}$, 481	$8.2\text{E-}5\ n^{1.04}$, 159	$5.9\text{E-}5\ n^{1.03}$, 95.8
	fpc3-search	$1.6\text{E-}4\ n^{1.04}$, 321	$8.0\text{E-}5\ n^{1.05}$, 158	$6.1\text{E-}5\ n^{1.03}$, 95.9
	gpsr-bbMono	$4.1\text{E-}4\ n^{1.04}$, $\gg$ 300	$2.2\text{E-}4\ n^{1.03}$, 356	$8.4\text{E-}5\ n^{1.03}$, 131
	stomp	not rec.	$6.4\text{E-}5\ n^{1.08}$, 199	$2.9\text{E-}5\ n^{1.15}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576, except for [a]. Then n = 524,288.

FPC and GPSR were run at $\mu = \mu_{\text{rec}}$, and their results were not de-biased.

DCT A, $\sigma_1 = 0$, σ_2 = 1E-2, $n_{\max} = 1{,}048{,}576$

		$\delta = 0.1$	$\delta = 0.3$	$\delta = 0.5$
$\rho = 0.3$	fpc3-basic		0.00053 $n^{1.02}$, 367[a]	6.4E-5 $n^{1.05}$, 146
	fpc3-search		0.00061 $n^{1.02}$, 395[a]	6.9E-5 $n^{1.05}$, 154
	gpsr-bbMono		0.0015 $n^{1.03}$, $\gg$ 300[a]	6.3E-5 $n^{1.03}$, 103
	stomp		not rec.	1.8E-4 $n^{1.09}$, $\gg$ 300
$\rho = 0.2$	fpc3-basic	6.3E-4 $n^{1.04}$, 555[a]	1.9E-4 $n^{1.03}$, 294	4.8E-5 $n^{1.04}$, 89.6
	fpc3-search	4.0E-4 $n^{1.02}$, 293[a]	1.6E-4 $n^{1.03}$, 267	3.7E-5 $n^{1.06}$, 94.6
	gpsr-bbMono	1.3E-3 $n^{1.04}$, $\gg$ 300[a]	6.6E-4 $n^{1.01}$, $\gg$ 300	3.1E-5 $n^{1.03}$, 47.3
	stomp	not rec.	1.7E-4 $n^{1.08}$, 519	5.8E-5 $n^{1.14}$, 431
$\rho = 0.1$	fpc3-basic	2.5E-4 $n^{1.04}$, 481	8.2E-5 $n^{1.04}$, 159	5.9E-5 $n^{1.03}$, 95.8
	fpc3-search	1.6E-4 $n^{1.04}$, 321	8.0E-5 $n^{1.05}$, 158	6.1E-5 $n^{1.03}$, 95.9
	gpsr-bbMono	4.1E-4 $n^{1.04}$, $\gg$ 300	2.2E-4 $n^{1.03}$, 356	8.4E-5 $n^{1.03}$, 131
	stomp	not rec.	6.4E-5 $n^{1.08}$, 199	2.9E-5 $n^{1.15}$, 263

All entries are CPU solve times, in s, unless the given algorithm was not able to recover x_s.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 1,048,576, except for [a]. Then n = 524,288.

FPC and GPSR were run at $\mu = \mu_{\text{rec}}$, and their results were not de-biased.

An Approximation for M

When there is signal noise and $AA^T \neq I$, **calculating M and μ can be very expensive**.

An alternative is to approximate

$$M \approx (\sigma_1^2 \bar{\sigma}^2 + \sigma_2^2) I,$$

where $\bar{\sigma}^2 = \lambda_{\max}(AA^T)$.

The next set of results investigate the advantages and disadvantages of this approximation.

Gaussian A, σ_1 =1E-2, $\sigma_2 =$ 1E-8, $n_{\max} = 4096$

		μ	M	$\delta = 0.3$ data	solve	rel. err.
$\rho = 0.1$	fpc3-basic	$\mu_{\text{rec}}/10$	full	3.3E-8$n^{2.62}$, 100	5.9E-6 $n^{1.45}$, 1.10	0.015
	fpc3-search	$\mu_{\text{rec}}/10$	full	6.5E-9$n^{2.82}$, 110	1.1E-5 $n^{1.36}$, 0.94	0.015
	gpsr-bbMono	$\mu_{\text{rec}}/10$	full	1.2E-8$n^{2.74}$, 110	5.8E-5 $n^{1.11}$, 0.65	0.015
	fpc3-basic	$\mu_{\text{rec}}/10$	approx.	9.8E-8$n^{1.79}$, 0.27	1.3E-6 $n^{1.76}$, 2.70	0.017
	fpc3-search	$\mu_{\text{rec}}/10$	approx.	2.0E-6$n^{1.42}$, 0.31	7.0E-6 $n^{1.48}$, 1.55	0.017
	gpsr-bbMono	$\mu_{\text{rec}}/10$	approx.	1.7E-6$n^{1.45}$, 0.29	1.2E-7 $n^{1.97}$, 1.38	0.017
	fpc3-basic	μ_{rec}	approx.	3.3E-6$n^{1.35}$, 0.27	1.5E-6 $n^{1.78}$, 3.79	0.028
	fpc3-search	μ_{rec}	approx.	1.5E-7$n^{1.74}$, 0.32	3.1E-6 $n^{1.62}$, 2.38	0.028
	gpsr-bbMono	μ_{rec}	approx.	6.3E-6$n^{1.26}$, 0.27	1.7E-5 $n^{1.53}$, 6.01	0.028
	stomp	N/A	N/A	2.0E-6$n^{1.45}$, 0.45	4.8E-10 $n^{3.00}$, 28.2	0.040

The first two columns are CPU data generation or solve times, in s. The last is the relative error at $n = 4096$, $\|x - x_s\|/\|x_s\|$.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 4096.

FPC and GPSR include time for calculating M and μ in "data". De-biasing is included in "solve" when $\mu = \mu_{\text{rec}}$.

Gaussian A, σ_1 =1E-2, σ_2 = 1E-8, n_{max} = 4096

		μ	M	$\delta = 0.3$ data	solve	rel. err.
$\rho = 0.1$	fpc3-basic	$\mu_{rec}/10$	full	3.3E-8$n^{2.62}$, 100	5.9E-6 $n^{1.45}$, 1.10	0.015
	fpc3-search	$\mu_{rec}/10$	full	6.5E-9$n^{2.82}$, 110	1.1E-5 $n^{1.36}$, 0.94	0.015
	gpsr-bbMono	$\mu_{rec}/10$	full	1.2E-8$n^{2.74}$, 110	5.8E-5 $n^{1.11}$, 0.65	0.015
	fpc3-basic	$\mu_{rec}/10$	approx.	9.8E-8$n^{1.79}$, 0.27	1.3E-6 $n^{1.76}$, 2.70	0.017
	fpc3-search	$\mu_{rec}/10$	approx.	2.0E-6$n^{1.42}$, 0.31	7.0E-6 $n^{1.48}$, 1.55	0.017
	gpsr-bbMono	$\mu_{rec}/10$	approx.	1.7E-6$n^{1.45}$, 0.29	1.2E-7 $n^{1.97}$, 1.38	0.017
	fpc3-basic	μ_{rec}	approx.	3.3E-6$n^{1.35}$, 0.27	1.5E-6 $n^{1.78}$, 3.79	0.028
	fpc3-search	μ_{rec}	approx.	1.5E-7$n^{1.74}$, 0.32	3.1E-6 $n^{1.62}$, 2.38	0.028
	gpsr-bbMono	μ_{rec}	approx.	6.3E-6$n^{1.26}$, 0.27	1.7E-5 $n^{1.53}$, 6.01	0.028
	stomp	N/A	N/A	2.0E-6$n^{1.45}$, 0.45	4.8E-10 $n^{3.00}$, 28.2	0.040

The first two columns are CPU data generation or solve times, in s. The last is the relative error at $n = 4096$, $\|x - x_s\| / \|x_s\|$.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 4096.

FPC and GPSR include time for calculating M and μ in "data". De-biasing is included in "solve" when $\mu = \mu_{rec}$.

Gaussian A, σ_1 =1E-2, $\sigma_2 =$ 1E-8, $n_{\max} = 4096$

				$\delta = 0.3$		
		μ	M	data	solve	rel. err.
$\rho = 0.1$	fpc3-basic	$\mu_{\text{rec}}/10$	full	3.3E-8$n^{2.62}$, 100	5.9E-6 $n^{1.45}$, 1.10	0.015
	fpc3-search	$\mu_{\text{rec}}/10$	full	6.5E-9$n^{2.82}$, 110	1.1E-5 $n^{1.36}$, 0.94	0.015
	gpsr-bbMono	$\mu_{\text{rec}}/10$	full	1.2E-8$n^{2.74}$, 110	5.8E-5 $n^{1.11}$, 0.65	0.015
	fpc3-basic	$\mu_{\text{rec}}/10$	approx.	9.8E-8$n^{1.79}$, 0.27	1.3E-6 $n^{1.76}$, 2.70	0.017
	fpc3-search	$\mu_{\text{rec}}/10$	approx.	2.0E-6$n^{1.42}$, 0.31	7.0E-6 $n^{1.48}$, 1.55	0.017
	gpsr-bbMono	$\mu_{\text{rec}}/10$	approx.	1.7E-6$n^{1.45}$, 0.29	1.2E-7 $n^{1.97}$, 1.38	0.017
	fpc3-basic	μ_{rec}	approx.	3.3E-6$n^{1.35}$, 0.27	1.5E-6 $n^{1.78}$, 3.79	0.028
	fpc3-search	μ_{rec}	approx.	1.5E-7$n^{1.74}$, 0.32	3.1E-6 $n^{1.62}$, 2.38	0.028
	gpsr-bbMono	μ_{rec}	approx.	6.3E-6$n^{1.26}$, 0.27	1.7E-5 $n^{1.53}$, 6.01	0.028
	stomp	N/A	N/A	2.0E-6$n^{1.45}$, 0.45	4.8E-10 $n^{3.00}$,28.2	0.040

The first two columns are CPU data generation or solve times, in s. The last is the relative error at $n = 4096$, $\|x - x_s\|/\|x_s\|$.

- Solve time = Cn^{α} curve fit to (n,solve time) data (averages over 5 runs).
- Solve time for n = 4096.

FPC and GPSR include time for calculating M and μ in "data". De-biasing is included in "solve" when $\mu = \mu_{\text{rec}}$.

Conclusions

- FPC, GPSR-BB and StOMP are all useful given the right situation.
- However, GPSR-BB is not robust to μ (convergence slows and tolerances must be changed as μ increases).
- And StOMP's thresholding schemes are difficult to untangle and apply appropriately in new situations.
- On the other hand, FPC is simple, general and robust.
- l1_ls is as accurate and as easy to use as FPC, but is much slower.
- FPC (and GPSR-BB, if μ is small enough) outperforms StOMP when measurements are noisy and n is large.

Conclusions, Cont.

- De-biasing generally improves the results if the desired solution is truly sparse, and nonzeros can be accurately distinguished.
- Line search is often helpful, but more so for smaller μ and closer to the phase transition line (less sparsity). The "sweet spot" changes some with matrix type.

Acknowledgements

- Joint work with Yin Zhang and Wotao Yin.
- Supported by NSF VIGRE Grant DSM-0240058.

References I

D. Donoho, Y. Tsaig, I. Drori, and J.-C. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. *Preprint*, 2006.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *Preprint*, 2007.

S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. A method for large-scale l_1-regularized least squares problems with applications in signal processing and statistics. *Manuscript*, 2007.

An Optimality Condition

The gradient of $\|Ax - b\|_M^2$ is

$$g(x) := A^T M(Ax - b). \tag{5}$$

x^* is a solution to (2) if and only if

$$\mu g_i(x^*) \begin{cases} = -1, & x_i^* > 0, \\ \in [-1, 1], & x_i^* = 0, \\ = 1, & x_i^* < 0. \end{cases} \tag{6}$$

M and μ

M is the inverse covariance matrix of $Ax_s - b$.

μ is estimated based on the normal distribution of $Ax_s - b$, the optimality condition for (2), and relationships between norms.

Case 0: In general $M = \left(\sigma_1^2 AA^T + \sigma_2^2 I\right)^{-1}$, and we require an estimate of $\underline{\sigma}^2 = \lambda_{\min}(M^{1/2}AA^TM^{1/2})$.

$$\mu = \frac{1}{\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}.$$

Case 1: $\sigma_1 = 0$. Then $M = I$, $\underline{\sigma}^2 = \lambda_{\min}(AA^T)$, and $\mu = \frac{1}{\sigma_2\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.

Case 2: $AA^T = I$. Then $M = I$, and $\mu = \frac{1}{\sqrt{\sigma_1^2 + \sigma_2^2}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.

M and μ

M is the inverse covariance matrix of $Ax_s - b$.

μ is estimated based on the normal distribution of $Ax_s - b$, the optimality condition for (2), and relationships between norms.

Case 0: In general $M = \left(\sigma_1^2 AA^T + \sigma_2^2 I\right)^{-1}$, and we require an estimate of $\underline{\sigma}^2 = \lambda_{\min}(M^{1/2}AA^TM^{1/2})$.

$$\mu = \frac{1}{\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}.$$

Case 1: $\sigma_1 = 0$. Then $M = I$, $\underline{\sigma}^2 = \lambda_{\min}(AA^T)$, and $\mu = \frac{1}{\sigma_2\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.

Case 2: $AA^T = I$. Then $M = I$, and $\mu = \frac{1}{\sqrt{\sigma_1^2 + \sigma_2^2}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.

M and μ

M is the inverse covariance matrix of $Ax_s - b$.

μ is estimated based on the normal distribution of $Ax_s - b$, the optimality condition for (2), and relationships between norms.

Case 0: In general $M = \left(\sigma_1^2 AA^T + \sigma_2^2 I\right)^{-1}$, and we require an estimate of $\underline{\sigma}^2 = \lambda_{\min}(M^{1/2}AA^TM^{1/2})$.

$$\mu = \frac{1}{\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}.$$

Case 1: $\sigma_1 = 0$. Then $M = I$, $\underline{\sigma}^2 = \lambda_{\min}(AA^T)$, and $\mu = \frac{1}{\sigma_2\underline{\sigma}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.

Case 2: $AA^T = I$. Then $M = I$, and $\mu = \frac{1}{\sqrt{\sigma_1^2 + \sigma_2^2}}\sqrt{\frac{n}{\chi^2_{1-\alpha,m}}}$.