Quarterly Report

Tom Kealy

December 18, 2014

1 Introduction

2 Models

2.1 Common Setup

We are considering a radio environment with a single primary user (PU) and a network of J =50 nodes collaboratively trying to sense and reconstruct the PU signal, either in a fully distributed manner (by local message passing), or by transmitting measurements to a fusion centre which then solves the linear system.

We are trying to sense and reconstruct a wideband signal, divided into L channels. We have a (connected) network of J ($= 50$) nodes placed uniformly at random within the square $[0,1] \times [0,1]$.

2.2 Frequency Only

We are considering a radio environment with a single primary user (PU) and a network of J =50 nodes collaboratively trying to sense and reconstruct the PU signal, either in a fully distributed manner (by local message passing), or by transmitting measurements to a fusion centre which then solves the linear system.

We are trying to sense and reconstruct a wideband signal, divided into L channels. We have a (connected) network of J ($= 50$) nodes placed uniformly at random within the square $[0,1] \times [0,1]$. They individually take measurements in the following way: by mixing the incoming analogue signal $x(t)$ with a mixing function $p_i(t)$ aliasing the spectrum. $x(t)$ is assumed to be bandlimited and composed of up to k uncorrelated transmissions over the L possible narrowband channels - i.e. the signal is k-sparse.

This process is repeated in parallel at each node (unrelated to N_{sig} so that each band in x

appears in baseband. The mixing functions - which are independent for each node - are required to be periodic, with period T_p. Since p_i is periodic it has Fourier expansion:

$$p_i(t) = \sum_{l=-\infty}^{\infty} c_{il} \exp\left(jlt\frac{2\pi}{T_p}\right) \tag{2.2.1}$$

The c_{il} are the Fourier coefficients of the expansion and are defined in the standard manner. The result of the mixing procedure in channel i is therefore xp_i, with Fourier transform:

$$X_i(f) = \int_{-\infty}^{\infty} x(t)\, p_i(t)\, dt \tag{2.2.2}$$

$$= \sum_{l=-\infty}^{\infty} c_{il} X(f - lf_p) \tag{2.2.3}$$

(insert the Fourier series for p_i, then exchange the sum and integral). The output of this mixing process then, is a linear combination of shifted copies of $X(f)$, with at most $\lceil f_N Y Q/f_p \rceil$ terms since $X(f)$ is zero outside it's support (we have assumed this Nyquist frequency exists, even though we never sample at that rate).

Once the mixing process has been completed the signal in each channel is low-pass filtered and sampled at a rate $f_s \geq f_p$. In the frequency domain this is a ideal rectangle function, so the output of a single channel is:

$$Y_i\left(e^{j2\pi f T_s}\right) = \sum_{l=-L_0}^{+L_0} \tag{2.2.4}$$

since frequencies outside of $[-f_2/2, f_s/2]$ will filtered out. L_0 is the smallest integer number of non-zero contributions in $X(f)$ over $[-f_2/2, f_s/2]$ - at most $\lceil f_N Y Q/f_p \rceil$ if we choose $f_s = f_p$. These relations can be written in matrix form as:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w} \tag{2.2.5}$$

where $\mathbf{y}$ contains the output of the measurement process, and $\mathbf{A}$ is a product matrix of the mixing functions, their Fourier coefficients, a partial Fourier Matrix, and a matrix of channel co-efficients. $\mathbf{x}$ is the vector of unknown samples of $x(t)$.

i.e. $\mathbf{A}$ can be written:

$$\mathbf{A}^{m\times L} = \mathbf{S}^{m\times L}\mathbf{F}^{L\times L}\mathbf{D}^{L\times L}\mathbf{H}^{L\times L} \tag{2.2.6}$$

The measurements $\mathbf{y}$ are transmitted to a Fusion Centre via a control channel. The system can then be solved by linear programming.

2.3 Joint Space and Frequency

We write the power spectral density (psd) of the sth transmitter as:

$$\phi_s = \sum_b \beta_{bs} \psi_b (f) \tag{2.3.7}$$

This model expresses in psd of the transmitter in a suitable basis - for example $\psi_b (f)$ could be zero everywhere except for the set of frequencies where $f = b$ i.e. ψ is a rectangular function with height β_{bs} and support f. Other candidates for ψ include splines (e.g. raised cosines), and complex exponentials.

Given this, the psd at the rth receiver is:

$$\phi_r = \sum_s g_{sr} \phi_s = \sum_s g_{sr} \sum_b \beta_{bs} \psi_b (f) \tag{2.3.8}$$

where

$$g_{sr} = \exp (-||x_r - x_s||_2^{\alpha}) \tag{2.3.9}$$

is the channel response between the sth transmitter and the rth reciver.

This model can be summarised using Kronecker products as follows:

Let $\tilde{G} = g_s r^T$, e_r, e_b be unit vectors i.e. they are 1 for the i^{th} receiver or frequency band respectively.

The received power at a receiver (when only a single transmitter is transmitting) can be written:

$$y_r = \left(e_r^T \bigotimes I_{n_b} \right) y \tag{2.3.10}$$

with,

$$y = \left(\tilde{G} \bigotimes I_{n_b} \right) \phi \tag{2.3.11}$$

Now, we have

$$\phi = e_s \bigotimes \phi_s \tag{2.3.12}$$

so,

$$y = \left(\tilde{G} \bigotimes I_{n_b} \right) \left(e_s \bigotimes \phi_s \right) \tag{2.3.13}$$

finally we have,

$$y_r = \left(e_r^T \bigotimes I_{n_b}\right)\left[\left(\tilde{G} \bigotimes I_{n_b}\right)\left(e_s \bigotimes \phi_s\right)\right] \tag{2.3.14}$$

$\beta_{bs} \in \mathbb{R}^{1\times n_b}$, $g_{sr} \in \mathbb{R}^{n_r \times n_s}$ and $\psi_{kb} \in 1 \times n_k n_b$ where n_k is the number of frequency bands (in this example $n_k = n_b$.

In the absence of knowledge of the location of the transmitters we introduce a grid of *candidate* locations, to make the above model linear. s now runs over the set of these candidate locations.

The problem of estimating the coefficients, β, from noisy observations $y = \phi_r + N(0,1)$ is now one that can be tackled by linear regression/convex optimisation.

3 Distributed Optimisation

3.0.1 ADMM

The alternating direction method of multipliers (ADMM), is an algorithm intended to blend the decomposability of the gradient ascent algorithm **??**, with the more general method of multipliers **??**, i.e. when the objective function is decomposable but the dual function is not differentiable. The algorithm solves problems of the form

$$\min_x f(x) + g(z) \tag{3.0.15}$$

$$\text{s.t } Ax + Bz = c \tag{3.0.16}$$

where f and g are assumed to be convex function with range in $\mathbb{R}$, $A \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{p\times m}$ are matrices (not assumed to have full rank), and $c \in \mathbb{R}^p$. We begin by illustrating a few examples, relevant to the type of problems encountered in signal processing.

Example 3.1. *The Basis Pursuit problem*

$$\begin{aligned} \underset{x}{\textit{minimize}} \quad & \|x\|_1 \\ \textit{subject to} \quad & Ax = b \end{aligned}$$

can be written in the ADMM form (3.0.16):

$$\begin{aligned} \underset{x}{\textit{minimize}} \quad & f(x) + \|z\|_1 \\ \textit{subject to} \quad & x - z = 0 \end{aligned}$$

where f is the indicator function of $\{x \in \mathbb{R}^n : Ax = b\}$.

Example 3.2. *Similarly the LASSO:*

$$minimise\ \frac{1}{2}\|Ax-b\|_2^2+\lambda\|x\|_1 \tag{3.0.17}$$

$\lambda>0$, *can be solved by the ADMM with* $f=\frac{1}{2}\|Ax-b\|_2^2$ *and* $g=\lambda\|x\|_1$.

The solution of (3.0.16) is:

$$x*=\inf\{f(x)+g(z):Ax+Bz=c\} \tag{3.0.18}$$

As previously we form the Lagrangian:

$$L_\rho(x,y)=f(x)+g(z)+y^T(Ax+Bz-c)+\frac{\rho}{2}\|Ax+Bz-c\|_2^2 \tag{3.0.19}$$

ADMM consists of the iterations:

$$x^{k+1}:=\arg\min xL_\rho\left(x,z^k,y^k\right) \tag{3.0.20}$$

$$z^{k+1}:=\arg\min zL_\rho\left(x^{k+1},z,y^k\right) \tag{3.0.21}$$

$$y^{k+1}:=y^k+\rho\left(Ax^{k+1}+Bz^{k+1}-c\right) \tag{3.0.22}$$

Note that this algorithm is similar to both the gradient ascent and method of multipliers from the last section: it has an x minimisation step, an z minimisation step and a dual variable update. I.e. the Lagrangian is updated simultaneously with respect to the two primal variables. Separating minimisation into two steps (as opposed to joint minimisation) is what allows for decomposition if f or g (or both) are separable.

3.1 Example: Consensus

Suppose we want to solve a problem such as:

$$\underset{x}{\text{minimize}}\quad\sum_i f_i(x)$$

this could arise in statistical computing where f_i would be the loss function for the i^th block of training data. We can write the problem for distributed optimisation as:

$$\begin{aligned}\underset{x}{\text{minimize}}\quad&\sum_i f_i(x_i)\\ \text{subject to}\quad&x_i-z=0\end{aligned}$$

where x_i are local variables (for example local to each node in a spectrum sensing) and $x_i - z = 0$ are the consensus constraints. Consensus and regularisation can be achieved by adding a regularisation term $g(z)$ - for example $g(z) = \lambda||x||_1$ corresponds to the LASSO, and the f_i would be $f_i = ||A_i x_i - b||_2^2$.

As per the previous sections, we form the Augmented Lagrangian:

$$L_\rho(x, y) = \sum_i^n \left(f_i(x_i) + y_i^T (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2 \right) \tag{3.1.23}$$

The ADMM iterations for this Lagrangian are:

$$x_i^{k+1} := \arg\min x_i \left(f_i(x_i) + y_i^{kT} (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2 \right) \tag{3.1.24}$$

$$z^{k+1} := \frac{1}{n} \sum_i^n \left(x_i^{k+1} + (1\rho) y_i^k \right) \tag{3.1.25}$$

$$y_i^{k+1} := y_i^k + \rho \left(x_i^{k+1} - z^{k+1} \right) \tag{3.1.26}$$

The z^{k+1} iteration is analytic as we're minimising the squared norm of $x_i - z$ - so we average. With $\|x\|_1$ regularisation we perform soft-thresholding after the z update.

At each iteration the sum of the dual variables y_i is zero, so the algorithm can be simplified to:

$$x_i^{k+1} := \arg\min x_i \left(f_i(x_i) + y_i^{kT} \left(x_i - \bar{x}^k \right) + \frac{\rho}{2} \|x_i - \bar{x}^k\|_2^2 \right) \tag{3.1.27}$$

$$y_i^{k+1} := y_i^k + \rho \left(x_i^{k+1} - z^{k+1} \right) \tag{3.1.28}$$

where

$$\bar{x}^k = \frac{1}{n} \sum_i^n x_i^k \tag{3.1.29}$$

This algorithm can be summarised as follows: in each iteration

- gather x^k and average to get $\bar{x}^k$
- scatter the average to nodes
- update y_i^k locally
- update x_i locally

Each agent is minimising it's own function, plus a quadratic term (the squared norm) which penalises the agent from moving too far from the previous average.

Note that the 'gather' stage doesn't require a central processor - this can be done in a distributed manner also.

3.2 Constrained Optimisation on Graphs

We model the network as an undirected graph $G = (V, E)$, where $V = \{1 \dots J\}$ is the set of vertices, and $E = V \times V$ is the set of edges. An edge between nodes i and j implies that the two nodes can communicate. The number of nodes node i can communicate is written $\mathcal{N}_i$ and the degree of node i is $D_i = |\mathcal{N}_i|$.

We assume that a proper (or approximate) colouring of the graph is available: that is each node is assigned a number from a set $C = \{1 \dots c\}$, and no node shares a colour with any neighbour.

Given the measurement model from the section (SECTION), we can represent the linear system by concatenating the measurements of each node into a single system. Where, each row of the matrix represents the measurements of a single node. I.e we are trying to solve the following problem:

$$\min ||x||_1 \text{ subject to } A_p x = b_p \tag{3.2.30}$$

The problem is coupled at each node by the variable x. To ease this issue, at each node create a local copy, denoted x_j (so there will be J copies of this variable).

Then, for consistency, we constrain the problem, so that each copy is identical: $x_1 = x_2 = \dots x_J$. Equivalently, given the graphical structure of the problem, we require that $x_i = x_j$ for each edge of the network.

So now we are solving the problem

$$\min \frac{1}{J} \sum_J ||x_j||_1 \text{ subject to } A_p x = b_p \text{ and } x_i = x_j \{i, j\} \in E \tag{3.2.31}$$

This is exactly the kind of problem that can be attacked by the Alternating Direction Method of Multipliers, described previously. We present a short example for bipartite graphs, which can easily be generalised to graphs with many colours.

3.3 Example: bipartite graphs

We are here considering graphs which are connected, and 2-colourable, that is nodes 1 to c have colour 1 (red say), and nodes $c + 1$ to J have colour 2 (blue). We will demonstrate that we can decompose problem (3.2.31) into c problems which can be executed in parallel by minimising x at

the 1st set of nodes, and $J-c$ problems which also can be executed in parallel by minimising at the second set of nodes.

Firstly, for compactness, we introduced the arc-incidence matrix of the graph B. This is the $J \times E$ matrix where each column corresponds to an edge $\{i,j\} \in E$, with the ith and jth entries equal to 1 and -1 respectively. With this notation, we can rewrite (3.2.31) as:

$$\min \frac{1}{J} \sum_{J} ||x_j||_1 \text{ subject to } A_p x = b_p \text{ and } \left(B^T \circ I\right) \bar{x} = 0 \tag{3.3.32}$$

where $\bar{x} = (x_1, x_2, \dots x_J)$, and $\circ$ is the Kronecker product.

In this form, the Lagrangian of the problem (3.3.32) can be written in the following way:

$$L\left(\bar{x}_1, \bar{x}_2; \lambda\right) = \frac{1}{J} \sum_{j \in C_1} ||x_j||_1 + \frac{1}{J} \sum_{j \in C_2} ||x_j||_1 + \tag{3.3.33}$$

$$\phi_1\left(\bar{x_1}, \lambda\right) + \phi_2\left(\bar{x_2}, \lambda\right) + \tag{3.3.34}$$

$$\rho \bar{x}_1^T \left(B_1 B_2^T \circ I_n\right) \bar{x}_2 \tag{3.3.35}$$

where

$$\phi_i\left(\bar{x}_i, \lambda\right) = \lambda^T \left(B_i^T \circ I_n\right) \bar{x}_i + \frac{\rho}{2} \left|\left|\left(B_i^T \circ I_n\right) \bar{x}_i\right|\right|^2 \tag{3.3.36}$$

$$= \left(\left(B_i^T \circ I_n\right) \lambda\right)^T \bar{x}_i + \frac{\rho}{2} \bar{x}_i^T \left(B_1 B_2^T \circ I_n\right) \bar{x}_i \tag{3.3.37}$$

Since, no nodes within C_i are neighbours $B_i B_i^T$ is a diagonal matrix (with the degree of node i as the ith entry). So, we can finally re-write

$$\phi_i\left(\bar{x}_i, \lambda\right) = \sum_{j \in C_i} \left(\left(\sum_{k \in \mathcal{N}_i} sign\left(k-i\right) \lambda_{\{i,k\}} \right)^T x_i + \frac{\rho}{2} D_i \left|\left|x_i\right|\right|^2 \right) \tag{3.3.38}$$

where the dual variable λ has been decomposed edge-wise so that $\lambda_{\{i,j\}} = \lambda_{\{j,i\}}$ and is associated with edge $\{i,j\}$ and the constraint $x_i = x_j$.

4 Results

4.1 Frequency only Model

The model described in section (2.2) was simulated, with a wideband signal of 201 channels and a network of 50 nodes (i.e. the signal will be sampled at a 1/4 of rate predicted by Nyquist theory).

The mixing patterns were generated from iid Gaussian sources (i.e the matrix S had each entry drawn from an iid Gaussian source). Monte Carlo simulations were performed at EbN0db values ranging from 5 to -5 in steps of 1, and the expected Mean Squared Error (MSE) of solutions of a centralised solver (spgl1) and a distributed solver (ADMM). The results can be seen in fig (4.1).

These results indicate that for both centralised and distributed solvers, adding noise to the system results in a degrading of performance. Interestingly note, that the distributed solver seems to (slightly) outperform the centralised solver at all SNRs. This is counter-intuitive, as it would be expected that centralised solvers knowing *all* the available information would outperform distributed solutions. We conjecture that the updates described in section 3.2, take into account differences in noise across the network. The distributed averaging steps, which form the new prior for each node, then penalise updates from relatively more noisy observations.

This observation is (partially) confirmed in figure (4.2), which plots the progress of the centralised and distributed solvers (as a function of iterations) towards the optimum solution. The SNR is 0.5 (i.e the signal is twice as strong as the noise).

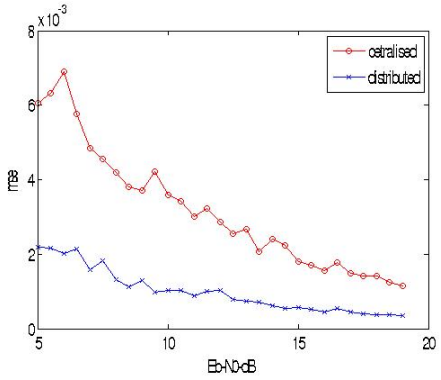

Figure 4.1: Mse vs SNR for the 'frequency only sensing model, showing the performance of distributed and centralised solvers

4.2 Joint Space-Frequency

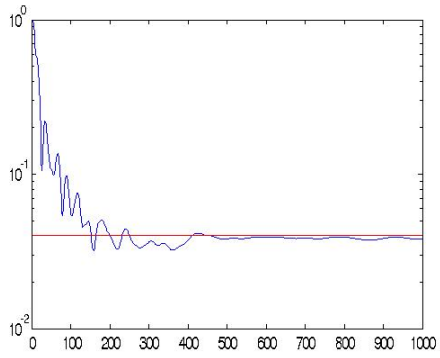

Figure 4.2: The progress of a distributed and a centralised solver as a function of the number of iterations

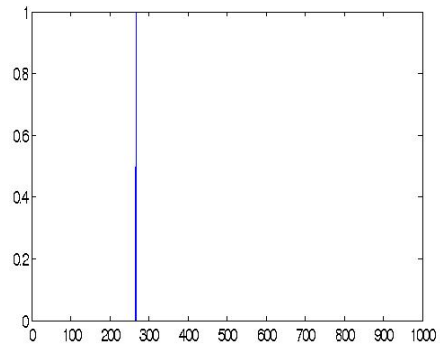

Figure 4.3: Example of β_{bs} for the 'joint space-frequency' model

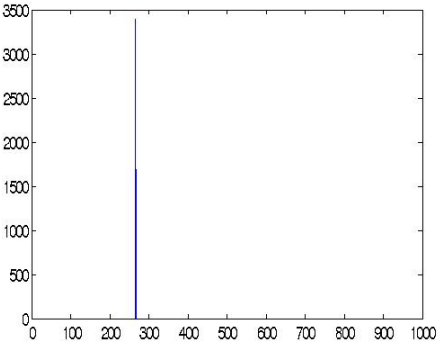

Figure 4.4: β_{bs} as seen by a node after a round o ADMM

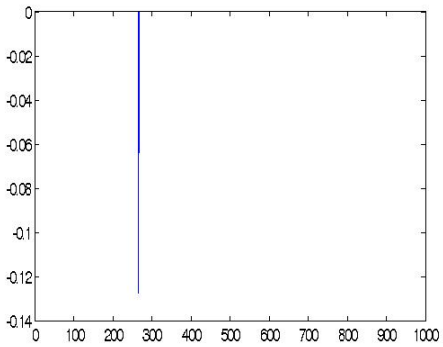

Figure 4.5: β_{bs} as produced by the centralised solver spgl1

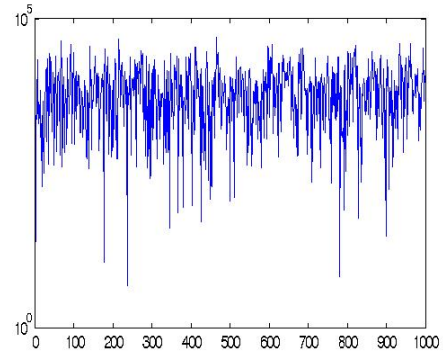

Figure 4.6: The progress of ADMM as a function of iterations for the joint model

5 Conclusions and Further Work

To conclude:

- We have extended the framework of paper (REF MOTA) to include joint space-frequency estimation as well as frequency estimation.
- Figure 4.6 suggests that we are over-fitting to the data, which could be the source of the issues
- We are able to successfully recover the support of the joint model, but not the full PSD at this time. This should be fairly easy to resolve.
- We conjecture the superior performance of distributed algorithms is a true phenomenon, and our statistical understanding of the ADMM algorithm allows us to begin explaining this.

  Improvements to the ADMM algorithm come in two forms:

  - Reducing the total number of rounds
  - Reducing the number of iterations per round

  Regarding the first point - since ADMM can be thought of as iterative MAP inference, we can reduce the number of iterations by more heavily penalising nodes with 'bad' iteration solutions. That is, we may be able to exploit the ability of the distributed algorithm to take account of variations in noise to drive the algorithm to a solution faster.

  Note that for ADMM, there is a required order of communication (in our case represented by a colouring of the network). Removing the need for a required order of communication may lead to fewer communications per iteration round.

.1 Statistical Interpretation

At each step k of the algorithm each agent is minimising it's own loss function, plus a quadratic. This has a simple interpretation: we're doing MAP estimation under the prior $\mathcal{N}\left(\bar{x}^k + (1\rho)\, y_i^k, \rho I\right)$. I.e. the prior mean is the previous iteration's consensus shifted by node i disagreeing with the previous consensus.

I think what's going on is that after some (large) number of steps $k*$ we can replace the prior iteration mean with the 'optimal' mean i.e. $|x_i^{k*} - \bar{x*}| \leq \varepsilon$, and taking a convex combination of these $x_j^{k*} \forall j \in \{1 \dots n\}$ we get a slightly better estimate because for any finite k large enough not all the x_j will be the same.

I think that when we add regularisation, we change the prior to a Laplace prior, with parameters corresponding to the previous iteration.

.2 ADMM for LASSO

As with the previous section ADMM can be formulated as the an iterative MAP estimation for the problem:

$$\frac{1}{2}\|Ax-b\|_2^2+\lambda\|x\|_1 \tag{.2.39}$$

The ADMM iterations are (in closed form):

$$x^{k+1} := \left(A^TA_\rho I\right)^{-1}\left(A^Tb+\rho\left(z^k-y^k\right)\right) \tag{.2.40}$$

$$z^{k+1} := S_{\lambda/\rho}\left(x^{k+1}+y^k/\rho\right) \tag{.2.41}$$

$$y^{k+1} := y^k+\rho\left(x^{k+1}-z^{k+1}\right) \tag{.2.42}$$

Where $S_{\lambda/\rho}(\circ)$ is the soft thresholding operator: $S_\gamma(x)_i = sign(x_i)\left(|x_i|-\gamma\right)^+$.

This algorithm has a nice statistical interpretation: it iteratively performs ridge regression, followed by shrinkage towards zero. Exactly like the MAP estimate for LASSO problem under a Laplace prior.

The soft-thresholding operator can be derived by considering the MAP estimate of the following model:

$$y = x+w \tag{.2.43}$$

and we seek

$$\hat{x} = \arg\max_x \mathbb{P}_{x|y}(x|y) \tag{.2.44}$$

This can be recast in the following form by using Bayes rule, noting that the denominator is independent of x and taking logarithms:

$$\hat{x} = \arg\max_x \left[\log\mathbb{P}_n(y-x)+\log\mathbb{P}_n(y-x)\right] \tag{.2.45}$$

The term $\mathbb{P}_n(y-x)$ arises because we are considering $x+n$ with n zero mean Gaussian. So, the conditional distribution of y (given x) will be a Gaussian centred at x.

We will take $\mathbb{P}_x(x)$ to be a Laplacian distribution:

$$\mathbb{P}_x(x) = \frac{1}{\sqrt{2}\sigma}\exp-\frac{\sqrt{2}}{\sigma}|x| \tag{.2.46}$$

Taking the maximum of .2.45 we obtain:

$$\frac{y-\hat{x}}{\sigma_n^2} - \frac{\sqrt{2}}{\sigma} = 0 \tag{.2.47}$$

Which leads the soft thresholding operation defined earlier, with $\gamma = \frac{\sqrt{2}\sigma_n^2}{\sigma}$

.3 ADMM for BPDN

The basis pursuit problem:

$$\min_x \left[\|x\|_1 \text{ s.t. } \|Ax-b\|_2 \leq \sigma\right] \tag{.3.48}$$

also has a closed form set of ADMM iterations:

Firstly, we write the program as:

$$\min_x \left[\|x\|_1 : Ax + y = b \|y\|_2 \leq \sigma\right] \tag{.3.49}$$

and the ADMM iterations are:

$$x^{k+1} := \Pi_{B_\sigma}\left(y^k/\rho - \left(Ay^k - b\right)\right) \tag{.3.50}$$

$$z^{k+1} := S_{\lambda/\rho}\left(x^{k+1} + y^k/\rho\right) \tag{.3.51}$$

$$y^{k+1} := y^k + \rho\left(x^{k+1} - z^{k+1}\right) \tag{.3.52}$$