This is just a quick document for describing the differences between vanilla gradient descent, iterative thresholding and AMP.

1 Gradient Descent

Suppose we observe a vector $y \in \mathbb{R}^m$, through an operator $A \in \mathbb{R}^{m \times n}$, i.e. we have the model

$$y = Ax \tag{1.0.1}$$

$x \in \mathbb{R}^n$. One way to guess x, is to minimise the mean squared error:

$$J = \frac{1}{2} ||y - Ax||_2^2 = (y - Ax)^T (y - Ax) \tag{1.0.2}$$

We can do this by gradient descent, as

$$\frac{\partial J}{\partial x} = A^T(y - Ax) \tag{1.0.3}$$

So we have an iterative algorithm:

$$x^{t+1} = x^t + A^T(y - Ax^t)$$

Putting it in a more suggestive form:

$$\begin{aligned} x^{t+1} &= x^t + A^T z^t \\ z^t &= y - Ax^t \end{aligned}$$

2 Iterative Thresholding

The algorithm from the previous section will perform poorly, as we've not used any prior information about x. We assume that x is sparse, so we now seek solutions to:

$$J = \frac{1}{2} ||y - Ax||_2^2 + \lambda ||x||_1 \tag{2.0.4}$$

The presence of the ℓ_1 norm, is inconvenient: J in this case has no formal gradient. In this case we can pretend that J is differentiable by doing a step in the direction of the gradient, and correcting for any error using the soft thresholding operator:$S_\gamma (x)_i = \text{sign}(x_i) (|x_i| - \gamma)^+$.

So our algorithm becomes:

$$\begin{aligned} x^{t+1} &= S_\lambda \left(x^t + A^T z^t\right) \\ z^t &= y - Ax^t \end{aligned}$$

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

$$y = x + w \tag{2.0.5}$$

where x is some (sparse) signal, and w is additive white Gaussian noise. We seek

$$\hat{x} = \arg\max_x \mathbb{P}_{x|y}(x|y) \tag{2.0.6}$$

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

$$\hat{x} = \arg\max_x \left[\log \mathbb{P}_w(y-x) + \log \mathbb{P}(x)\right] \tag{2.0.7}$$

The term $\mathbb{P}_n(y-x)$ arises because we are considering $x+w$ with w zero mean Gaussian, with variance σ_n^2. So, the conditional distribution of y (given x) will be a Gaussian centred at x.

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

$$\mathbb{P}(x) = \frac{1}{\sqrt{2}\sigma} \exp -\frac{\sqrt{2}}{\sigma}|x| \tag{2.0.8}$$

Note that $f(x) = \log \mathbb{P}_x(x) \ -\frac{\sqrt{2}}{\sigma}|x|$, and so by differentiating $f'(x) = -\frac{\sqrt{2}}{\sigma}\text{sign}(x)$

Taking the maximum of 2.0.7 we obtain:

$$\frac{y-\hat{x}}{\sigma_n^2} - \frac{\sqrt{2}}{\sigma} sign(x) = 0 \tag{2.0.9}$$

Which leads the soft thresholding operation defined earlier, with $\gamma = \frac{\sqrt{2}\sigma_n^2}{\sigma}$ as (via rearrangement):

$$y = \hat{x} + \frac{\sqrt{2}\sigma_n^2}{\sigma}\text{sign}(x)$$

or

$$\hat{x}(y) = \text{sign}(y)\left(y - \frac{\sqrt{2}\sigma_n^2}{\sigma}\right)_+$$

i.e $S_\gamma(y)$.

3 AMP

Approximate message passing further corrects this idea of pretending our optimisation objective is differentiable and correcting, by making a quadratic approximation to the likelihood of the model.

The algorithm becomes

$$\begin{aligned} x^{t+1} &= S_\lambda\left(x^t + A^T z^t\right) \\ z^t &= y - Ax^t + b_t z^t \end{aligned}$$

which is similar to the iterative thresholding algorithm, but with an additional 'momentum' term added. A good choice of b is:

$$\frac{1}{m}\left|\left|x^t\right|\right|_0 \tag{3.0.10}$$

where $||t||_0$ is the number of non-zero elements of t.

4 DISTA

procedure DADMM(y_j, M_j, ε)
  $x^0 = 0$, $z^0 = 0$, $\theta^0 = 0$, $\eta^0 = 0$,
$Q = \left(M_j^T M_j + (\rho D_J + 1)I\right)^{-1}$, $w_j = M_j^T y_j$
  while $\left|\left|z^{k+1} - z^k\right|\right| \leq \varepsilon$ **do**
    for $c = 1, \ldots, C$ **do**
      $x^{k+1} \leftarrow Q\left(w_j + z^k - \theta^{kT} - \nu^{kT}\right)$
      $z^{k+1} \leftarrow S_{\beta/\rho}\left(x_j^{k+1}\right)$
      $\theta^{k+1} \leftarrow \theta_j^k + \rho\left(x^{k+1} - z^{k+1}\right)$
    end for
Each node transmits x^{k+1} in $\mathcal{N}_j$ and calculates
    $\nu_j^{k+1} \leftarrow \nu_j^k + \rho\left(\sum_{m \in N_j} z_m^k - z_j^k\right)$
  end while
  return z^{k+1}
end procedure

Figure 4.1: DADMM at Node j

procedure DAMP(y_j, M_j, ε)
  $x^0 = 0$, α,

  while $\left|\left|x_j^{k+1} - x_j^k\right|\right| \leq \varepsilon$ **do**
Each node transmits x^k in $\mathcal{N}_j$ and calculates
    $x_j^{k+1/2} \leftarrow \frac{1}{D_p+1)} \sum_{j \in N_j} x_j^k$
    for $c = 1, \ldots, C$ **do**
      $x^{k+1} \leftarrow x_j^{k+1/2} - \alpha S_\beta\left(M_j^T x_j^{k+1/2}\right)$
    end for
  end while
  return x^{k+1}
end procedure

Figure 4.2: DAMP at Node j

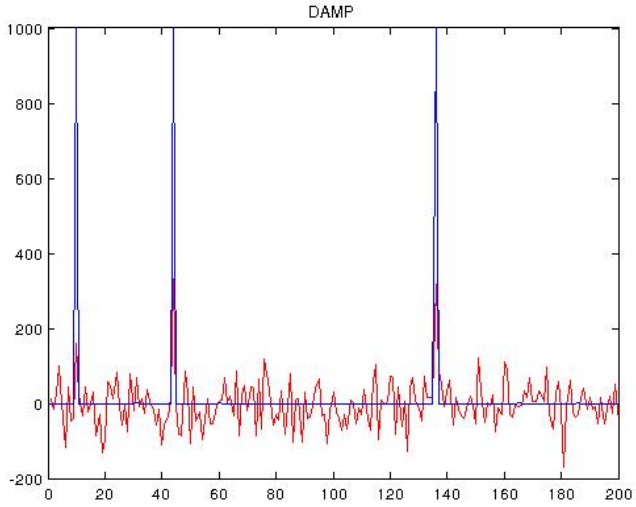

Figure 4.3: DAMP estimate after 1000 iterations

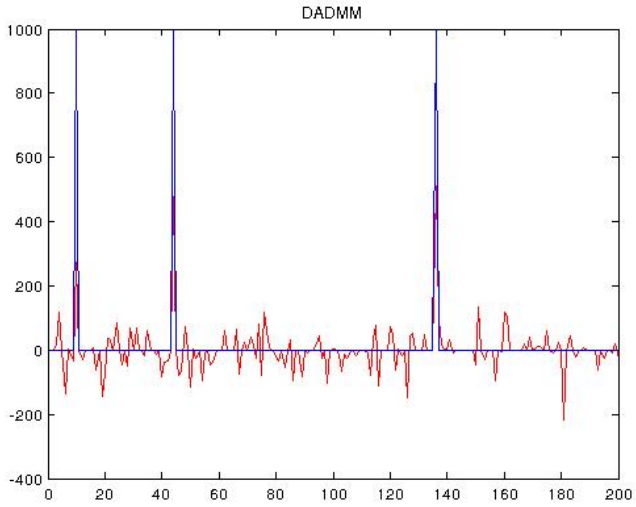

Figure 4.4: DADMM estimate after 1000 iterations

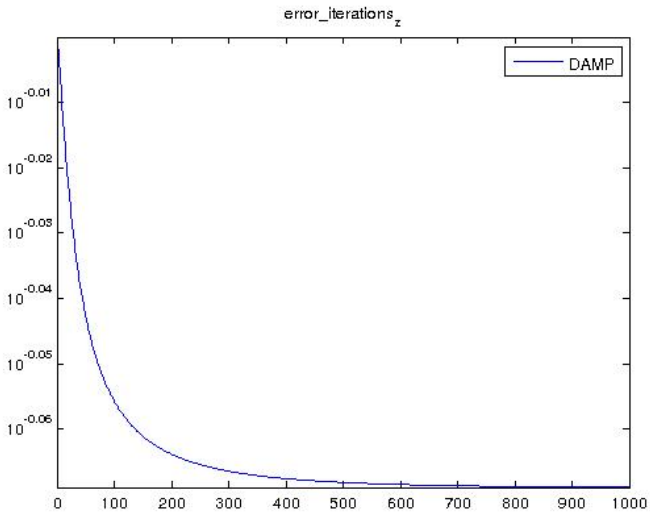

Figure 4.5: DAMP progress

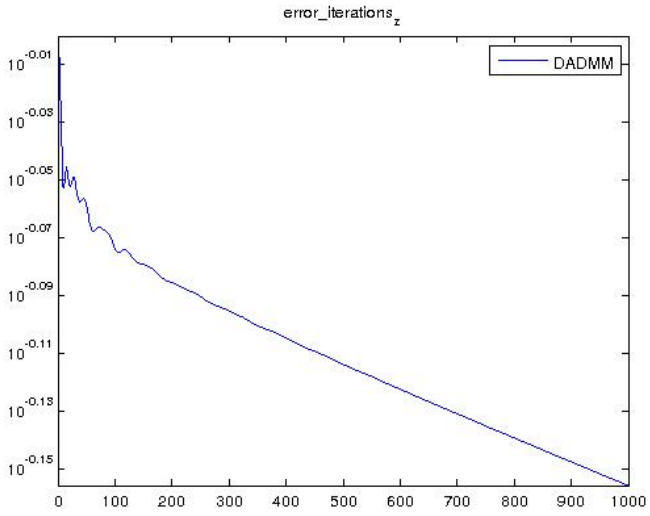

Figure 4.6: DADMM progress

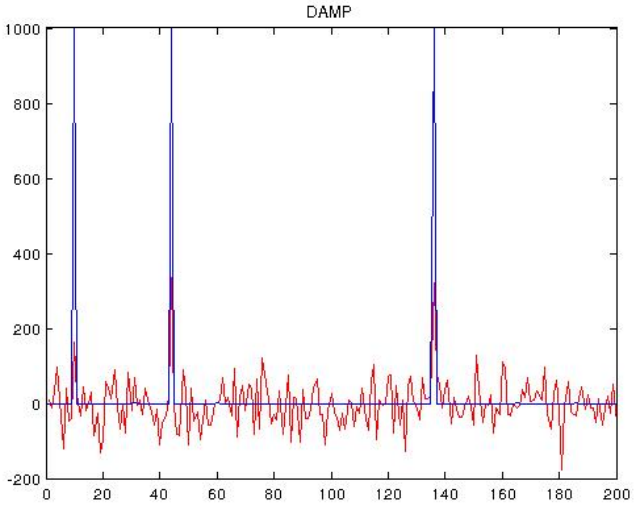

Figure 4.7: DAMP estimate after 5000 iterations

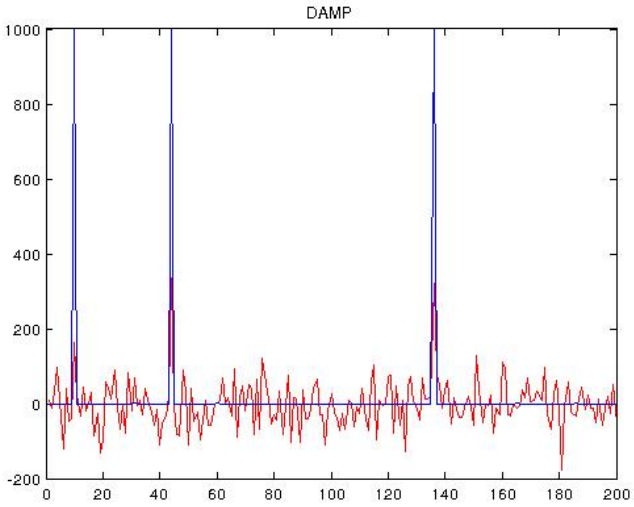

Figure 4.8: DADMM estimate after 5000 iterations

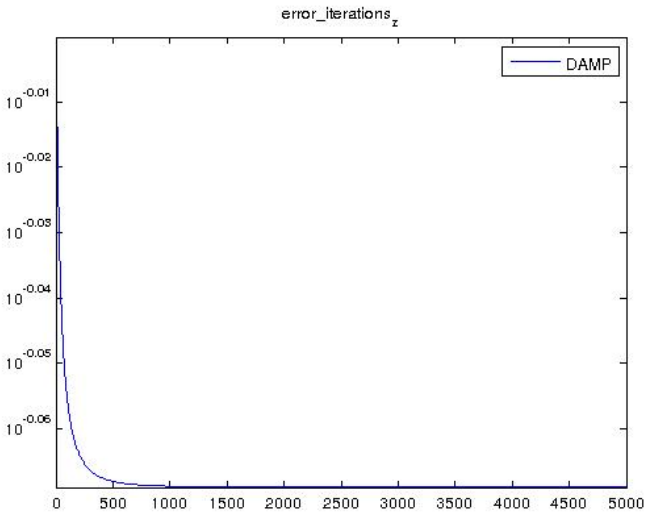

Figure 4.9: DAMP progress

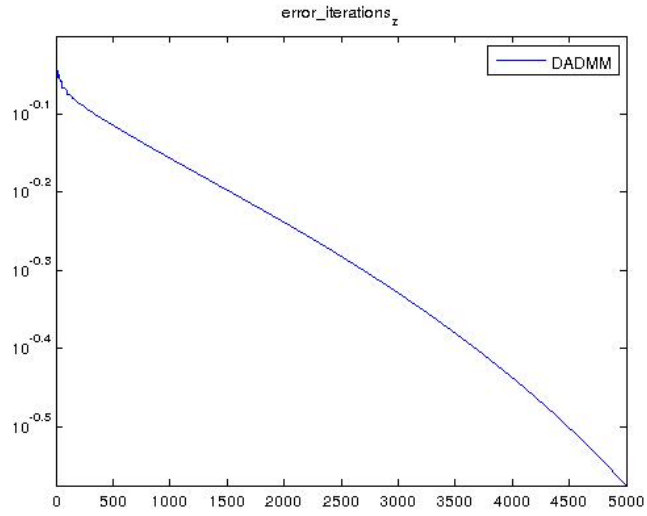

Figure 4.10: DADMM progress