Penalty and Shrinkage Functions for Sparse Signal Processing

Ivan Selesnick
Polytechnic Institute of New York University
selesi@poly.edu
November 5, 2012
Last edit: June 24, 2013

1 Introduction

The use of sparsity in signal processing frequently calls for the solution to the minimization problem

$$\arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{1}$$

where

$$\mathbf{x} = [x(1), \ldots, x(N)]^T,$$

$$\mathbf{y} = [y(1), \ldots, x(M)]^T,$$

$\mathbf{A}$ is a matrix of size $M \times N$, and $\|\mathbf{x}\|_2^2$ denotes the energy of of $\mathbf{x}$, defined as

$$\|\mathbf{x}\|_2^2 = \sum_n |x(n)|^2, \tag{2}$$

and $\|\mathbf{x}\|_1$ denotes the ℓ_1 norm of $\mathbf{x}$, defined as

$$\|\mathbf{x}\|_1 = \sum_n |x(n)|. \tag{3}$$

One approach in sparse signal processing calls for the solution to the more general problem:

$$\arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \sum_n \phi(x(n)). \tag{4}$$

The function $\phi(\cdot)$ is referred to as the penalty function (or regularization function). If $\phi(x) = \lambda|x|$, then (4) is the same as (1). For sparse signal processing, $\phi(x)$ should be chosen so as to promote sparsity of $\mathbf{x}$. It is common to set $\phi(x) = \lambda|x|$, especially because it is a convex function unlike many other sparsity promoting penalty functions. Convex functions are attractive because they can be more reliably minimized than non-convex functions. However, non-convex penalty functions can lead to enhanced sparsity of sparse signals.

MATLAB software online: http://eeweb.poly.edu/iselesni/lecture_notes/sparse_penalties/.

Several questions arise:

How should the penalty function $\phi(\cdot)$ be chosen?

What is the influence of the penalty function?

How can the function (4) be minimized with respect to $\mathbf{x}$?

2 Scalar Case

To better understand the role of the penalty function, it is informative to consider the scalar case. That is, given $y \in \mathbb{R}$, find $x \in \mathbb{R}$ so as to minimize the function

$$F(x) = \frac{1}{2}(y - x)^2 + \phi(x).$$

To emphasize the dependence of the minimizing value x on y, we define

$$s(y) := \arg\min_x \ \frac{1}{2}(y - x)^2 + \phi(x).$$

The function $s(y)$ is called the *shrinkage* function because it shrinks y towards zero. The exact form of $s(y)$ depends on the penalty function $\phi(x)$. Two particular examples of $\phi(x)$ will be analyzed in this section.

There are many other penalty functions that have been used for sparse signal/image processing [4, 5, 11, 14] (see Table 2 in [5] or Table 1 in [4]).

But these are two common, representative examples.

Example 1: Define the penalty function $\phi_1(x)$ as

$$\phi_1(x) = \lambda|x| \tag{5}$$

where $\lambda > 0$, as illustrated in Fig. 1. Note that $\phi_1(x)$ is a convex function. This corresponds to the ℓ_1 norm.

Example 2: Define

$$\phi_2(x) = \frac{\lambda}{a}\log(1 + a|x|) \tag{6}$$

where $\lambda > 0$ and $a > 0$, as illustrated in Fig. 2. Note that $\phi_2(x)$ is not a convex function.

For both of these two examples, we will find the shrinkage function $s(y)$.

To find the shrinkage function $s(y)$, we minimize $F(x)$ with respect to x. The derivative of $F(x)$ is given by

$$\frac{dF(x)}{dx} = x - y + \phi'(x).$$

Setting the derivative of $F(x)$ to zero gives

$$\boxed{y = x + \phi'(x)} \tag{7}$$

The value x minimizing $F(x)$ can be found by solving (7). First we need $\phi'(x)$.

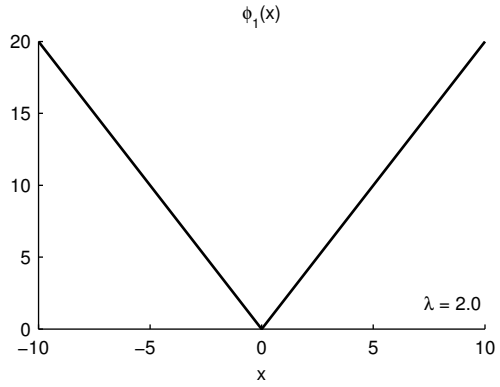

Figure 1: Penalty function $\phi_1(x) = \lambda|x|$.

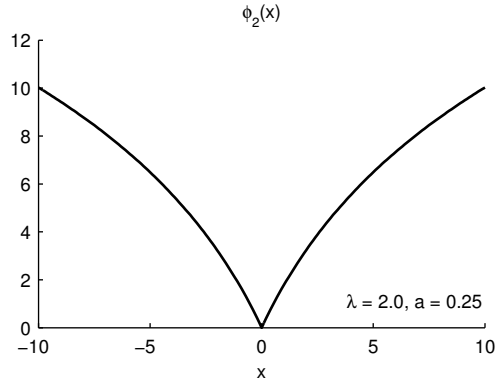

Figure 2: Penalty function $\phi_2(x) = (\lambda/a)\,\log(1 + a|x|)$.

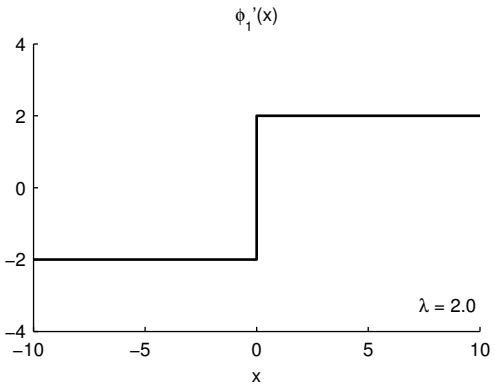

Figure 3: Derivative of penalty function $\phi_1'(x) = \lambda \, \mathrm{sign}(x)$.

Example 1: The derivative of $\phi_1(x)$ is

$$\phi_1'(x) = \begin{cases} \lambda, & x > 0 \\ -\lambda, & x < 0, \end{cases} \tag{8}$$

which is illustrated in Fig. 3. Note that $\phi_1(x)$ is not differentiable at $x = 0$ and we have omitted defining $\phi_1'(x)$ at $x = 0$. In fact, there is a mathematically rigorous method to define the derivative using convex analysis. (In convex analysis, the sub-differential is a generalization of the derivative that is defined as a *set*-valued function.) However, here we defer convex analysis and simply graph $\phi_1'(x)$ using a straight vertical line at $x = 0$ as illustrated in Fig. 3.

The function $\phi_1'(x)$ can be written compactly as

$$\phi_1'(x) = \lambda \, \mathrm{sign}(x) \tag{9}$$

where the function $\mathrm{sign}(x)$ is defined as

$$\mathrm{sign}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0. \end{cases} \tag{10}$$

Example 2: The derivative of $\phi_2(x)$ is given by

$$\phi_2'(x) = \begin{cases} \dfrac{\lambda}{1 + ax}, & x > 0 \\ -\dfrac{\lambda}{1 - ax}, & x < 0 \end{cases} \tag{11}$$

which can be written compactly as

$$\phi_2'(x) = \frac{\lambda}{1 + a|x|} \, \mathrm{sign}(x). \tag{12}$$

As in Example 1, the function $\phi_2(x)$ is not differentiable at $x = 0$. In the graph of $\phi_2'(x)$ shown in Fig. 4, we simply use a straight vertical line at $x = 0$.

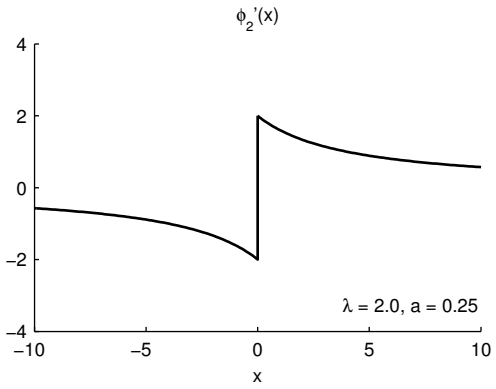

Figure 4: Derivative of penalty function $\phi_2'(x) = \lambda/(1 + a|x|)\,\mathrm{sign}(x)$.

Now that $\phi'(x)$ is found, we obtain the shrinkage function $s(y)$ by solving (7) for x.

Example 1: To find the shrinkage function, write

$$y = x + \phi_1'(x) \tag{13}$$

$$y = x + \lambda\,\mathrm{sign}(x). \tag{14}$$

Solving (14) for x can be done graphically. Fig. 5a shows the functions x and $\lambda\,\mathrm{sign}(x)$ individually. Fig. 5b shows the function $y = x + \lambda\,\mathrm{sign}(x)$. Solving for x entails simply exchanging the x and y axes of the graph in Fig. 5b. The shrinkage function $s_1(y)$ is given by the graph of x versus y in Fig. 5c. From the graph, we can write

$$s_1(y) = \begin{cases} y - \lambda, & y \geq \lambda \\ 0, & |y| \leq \lambda \\ y + \lambda, & y \leq -\lambda \end{cases}$$

which can be written compactly as

$$s_1(y) = \max(|y| - \lambda,\, 0)\ \mathrm{sign}(y).$$

The function $s_1(y)$ is known as the *soft threshold* function. It arises frequently in sparse signal processing and compressed sensing. It sets small values (less than λ in absolute value) to zero; and it shrinks large values (greater than λ in absolute value) toward zero.

Note that the soft threshold function is continuous and it has a maximum derivative of 1. Therefore, small changes in y do not produce large changes in $x = s_1(y)$. This is in contrast with shrinkage functions produced using some other penalty functions $\phi(x)$.

Example 2: To find the shrinkage function, write

$$y = x + \phi_2'(x) \tag{15}$$

$$y = x + \frac{\lambda}{1 + a|x|}\,\mathrm{sign}(x). \tag{16}$$

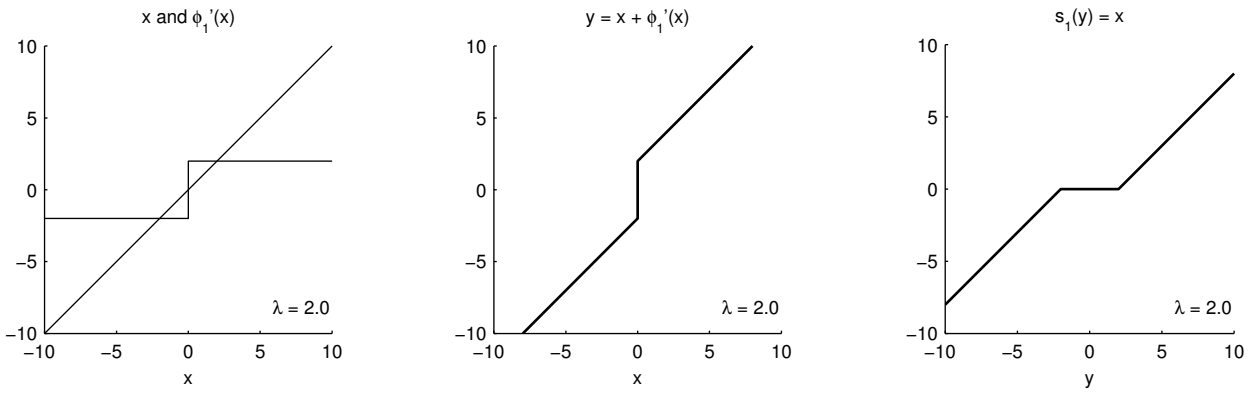

Figure 5: Derivation of the shrinkage function $s_1(y)$ for Example 1.

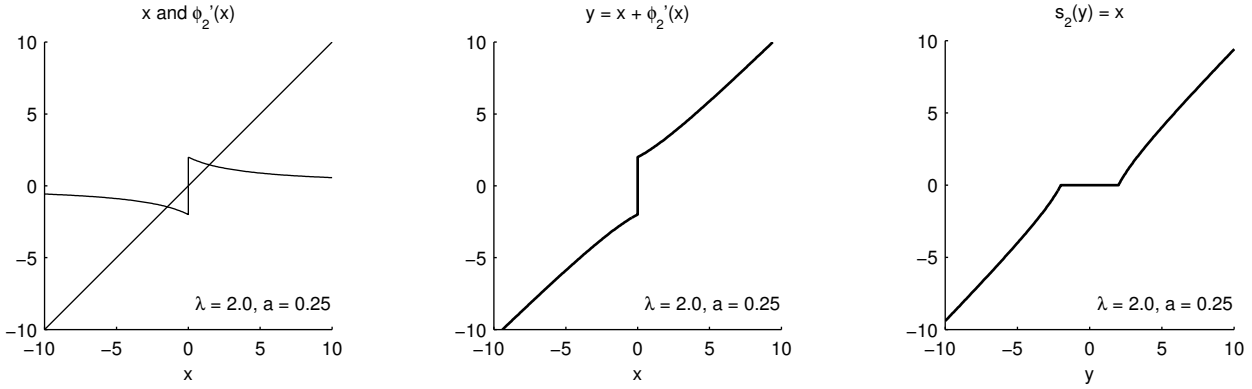

Figure 6: Derivation of the shrinkage function $s_2(y)$ for Example 2.

As for Example 1, solving (16) for x can be done graphically. Figures 6a and 6b show the functions x, $\phi_2'(x)$ individually, and their sum. The shrinkage function $s_2(y)$, obtained by exchanging the x and y axes of the graph in Fig. 6b, is shown in Fig. 6c.

A mathematical equation for $s_2(y)$ can be found by solving (16) for x. For $y \geq \lambda$, (16) is written as

$$y = x + \frac{\lambda}{1 + ax}$$

from which we write

$$y - x = \frac{\lambda}{1 + ax} \tag{17}$$

$$(y - x)(1 + ax) = \lambda \tag{18}$$

$$ax^2 + (1 - ay)x + (\lambda - y) = 0. \tag{19}$$

Using the quadratic formula gives

$$x = \frac{1}{2a}\left((ay - 1) + \sqrt{(ay - 1)^2 - 4a(\lambda - y)}\right), \qquad y \geq \lambda$$

A complete formula for the shrinkage formula $s_2(y)$ is given by

$$s_2(y) = \begin{cases} \frac{1}{2a}\left((a|y|-1) + \sqrt{(a|y|-1)^2 - 4a(\lambda - |y|)}\right) \text{ sign}(y), & |y| \geq \lambda \\ 0, & |y| \leq \lambda \end{cases} \tag{20}$$

Like the soft-threshold function, it sets small values (less than λ) to zero; and it shrinks large values (greater than λ) toward zero. But compared with the soft-threshold rule, large values are not as attenuated.

For both examples, we see that the shrinkage function sets all values of y less than λ to zero. Both shrinkage functions are *thresholding* functions. The *threshold* value is λ. But $\phi_2(x)$ has an additional parameter, a, which also affects the shape of the shrinkage function. This parameter can be used to adjust the shrinkage function.

How does a influence the shape of the shrinkage function $s_2(y)$?

First, we note that some values of a will lead to a complication in the derivation of the shrinkage function. In particular, when a is too large, then the graphs in Fig. 6 take on different characteristics. For example, with $\lambda = 2$ and $a = 3$, the new graphs are shown in Fig. 7. Note that the graph of $x + \phi_2'(x)$ in Fig. 7b is not a strictly increasing function of x. The graph shown in Fig. 7c does not define a function of y. There is not a unique x for each y.

Inspection of $F(x)$ reveals that two of the three values of x on the graph (where the graph provides non-unque x) are only *local* maxima and minima of $F(x)$. Only one of the three values is the global minimizer of $F(x)$. Taking into consideration the value of $F(x)$ so as to identify the minimizing x as a function of y, we obtain the shrinkage function illustrated in Fig. 8. (The darker line in the graph represents the shrinkage function.)

The shrinkage function in Fig. 8 is discontinuous. Some frequently used shrinkage functions are discontinuous like this (for example, the *hard threshold* function). However, sometimes a discontinuous shrinkage function is considered undesirable. A small change in y can produce a large change in $s(y)$. Such a shrinkage function is sensitive to small changes in the data, y.

To ensure the shrinkage function $s_2(y)$ is continuous, how should a be chosen?

Note that the discontinuity of the shrinkage function is due to $x + \phi'(x)$ not being a strictly increasing function of x. To avoid the discontinuity, $x + \phi'(x)$ should be increasing, i.e.

$$\frac{d}{dx}(x + \phi'(x)) > 0 \quad \text{for all } x > 0. \tag{21}$$

Equivalently, the penalty function $\phi(y)$ should satisfy

$$\phi''(x) > -1 \quad \text{for all } x > 0. \tag{22}$$

Note that, in Example 2,

$$\phi_2''(x) = -\frac{\lambda a}{(1+ax)^2}, \quad x > 0 \tag{23}$$

which is most negative at $x = 0^+$, at which point we have

$$\phi_2''(0^+) = -\lambda\, a. \tag{24}$$

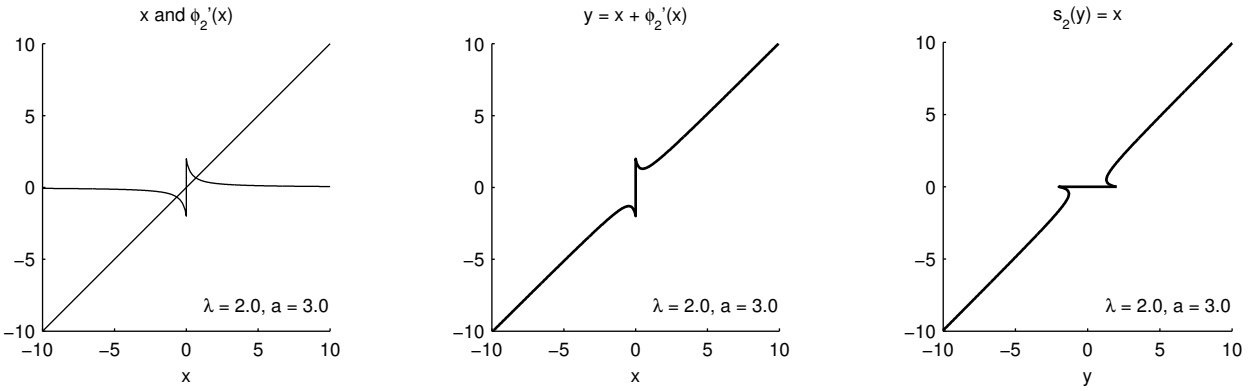

Figure 7: An issue arises in the derivation of the shrinkage function when $x + \phi'(x)$ is not strictly increasing.

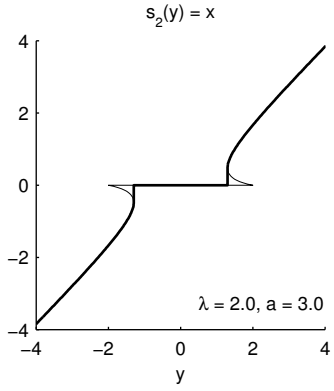

Figure 8: The shrinkage function $s_2(y)$ is discontinuous when $a \geq 1/\lambda$.

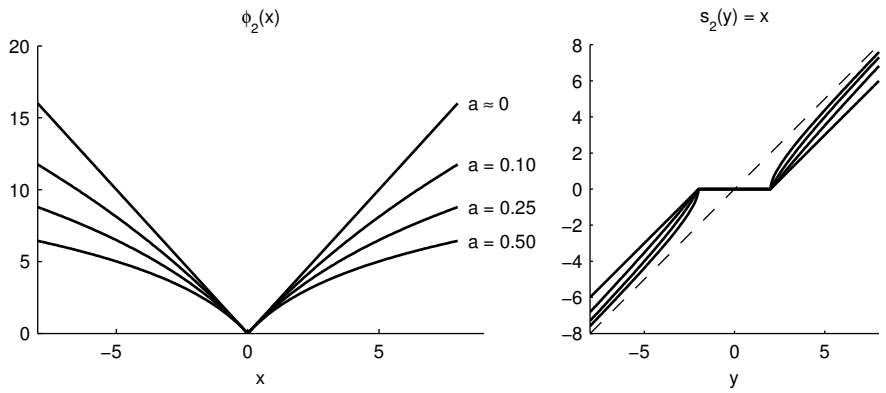

Figure 9: The penalty function $\phi_2(x)$ and shrinkage function $s_2(y)$ for $\lambda = 2$ and $a = 0, 0.1, 0.25, 0.5$.

From (22) and (23), we obtain the condition $-\lambda a > -1$ or $a < 1/\lambda$. Recalling that $a > 0$, we obtain the range

$$0 < a < \frac{1}{\lambda}. \tag{25}$$

Regarding the sensitivity of the shrinkage function — note that the maximum value of derivative of the shrinkage function $s_2(y)$ occurs at $y = \lambda$. Let us calculate $s_2'(\lambda^+)$. It can be found in two ways, by differentiating the shrinkage function, or indirectly as the reciprocal of the right-sided derivative of $x + \phi'(x)$ at $x = 0$. For Example 2, we have

$$s_2'(\lambda) = \frac{1}{1 - \lambda\, a} \tag{26}$$

so a can be set as

$$a = \frac{1}{\lambda}\left(1 - \frac{1}{s_2'(\lambda^+)}\right) \tag{27}$$

where $s_2'(\lambda^+)$ is the maximum slope of the shrinkage function.

The penalty function and shrinkage function for several values of a are shown in Fig. 9. In the limit as a approaches zero, the shrinkage function $s_2(y)$ approaches the soft-threshold rule, which has a slope equal to 1 for all $|y| > \lambda$.

3 From Shrinkage to Penalty Functions

In the preceding section, it was shown how the shrinkage function $s(y)$ can be derived from the penalty function $\phi(x)$. In this section, the reverse problem is considered. Given a shrinkage function $s(y)$; how can the corresponding penalty function $\phi(t)$ be derived? For example, consider the shrinkage function in Example 3.

Example 3: Define the shrinkage function

$$s_3(y) = \begin{cases} y - \dfrac{\lambda^2}{y}, & |y| \geq \lambda \\ 0, & |y| \leq \lambda \end{cases} \tag{28}$$

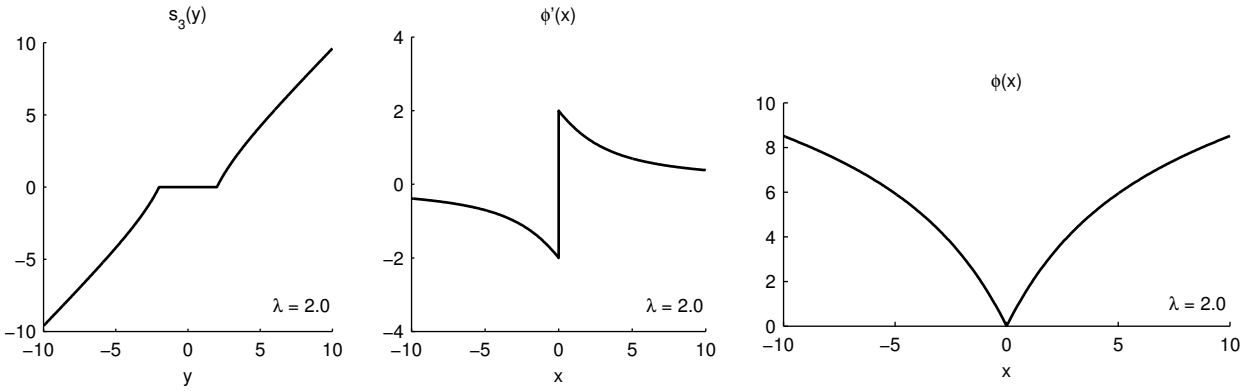

Figure 10: The shrinkage function $s_3(y)$ in (28), and the corresponding functions $\phi_3'(x)$ and $\phi_3(x)$.

as illustrated in Fig. 10. This shrinkage is sometimes referred to as the 'nonnegative garrote' [3,9,10]. The graph of the shrinkage function is similar to that in Fig. 6c, however, the formula is simpler; compare (28) with (20).

How can we find a penalty function $\phi(x)$ corresponding to the shrinkage function $s(y)$?

From (7) we have $y = x + \phi'(x)$ where $x = s(y)$. Hence, $\phi'(x)$ can be found by solving $x = s(y)$ for y and equating with $x + \phi'(x)$. For the shrinkage function in (28), for $y \geq \lambda$,

$$x = y - \frac{\lambda^2}{y} \tag{29}$$

$$y^2 - yx - \lambda = 0.$$

Using the quadratic formula gives

$$y = \frac{x}{2} + \frac{1}{2}\sqrt{x^2 + 4\lambda^2}. \tag{30}$$

Equating (30) with $x + \phi'(x)$ gives

$$\frac{x}{2} + \frac{1}{2}\sqrt{x^2 + 4\lambda^2} = x + \phi'(x) \tag{31}$$

or

$$\phi'(x) = \frac{1}{2}\sqrt{x^2 + 4\lambda^2} - \frac{x}{2}, \quad x > 0. \tag{32}$$

A complete formula for $\phi'(x)$ is

$$\phi_3'(x) = \frac{1}{2}\left(\sqrt{x^2 + 4\lambda^2} - |x|\right) \operatorname{sign}(x), \tag{33}$$

as illustrated in Fig. 10b.

For large x, the expression for $\phi'(x)$ in (33) involves the difference between two large numbers, which can be numerically inaccurate. The numerical calculation of $\phi'(x)$ for large x can be made more accurate, and its asymptotic behavior can be made more clear, using the identity

$$\left(\sqrt{x^2 + 4\lambda^2} - x\right)\left(\sqrt{x^2 + 4\lambda^2} + x\right) = 4\lambda^2 \tag{34}$$

which is verified by multiplying the terms on the left-hand-side. Using (34), the expression for $\phi'(x)$ in (33) can be written as

$$\phi_3'(x) = \frac{2\lambda^2}{\sqrt{x^2+4\lambda^2}+|x|}\,\mathrm{sign}(x). \tag{35}$$

From (35), it can be seen that $\phi_3'(x) \approx \lambda^2/x$ for large x.

The penalty function $\phi(x)$ can be obtained by integration,

$$\phi(x) = \int_0^{|x|} \phi'(\alpha)\, d\alpha \tag{36}$$

which yields

$$\phi_3(x) = \lambda^2 \,\mathrm{asinh}\left(\frac{|x|}{2\lambda}\right) + \frac{1}{4}|x|\left(\sqrt{x^2+4\lambda^2}-|x|\right) \tag{37}$$

$$= \lambda^2 \,\mathrm{asinh}\left(\frac{|x|}{2\lambda}\right) + \frac{\lambda^2|x|}{\sqrt{x^2+4\lambda^2}+|x|} \tag{38}$$

where $\mathrm{asinh}(z)$ is the inverse hyperbolic sine,

$$\mathrm{asinh}(z) = \log(z+\sqrt{z^2+1}). \tag{39}$$

The cost function $\phi_3(x)$ is illustrated in Fig.10c.

3.1 Comparison

Figure 11 shows the penalty functions and shrinkage functions for each of the three examples on the same axis. For each example, the parameter λ is set to 2, hence each shrinkage function exhibits a threshold value of 2. For the log penalty function (Example 2), the parameter a is set so that the (right-sided) derivative of the shrinkage function at $y = 2$ is equal to the slope of nn-garrote shrinkage function at that point. Namely, both the log and nn-garrote shrinkage functions have a (right-sided) derivative of 2 at $y = 2$; which is emphasized by the dashed line with slope 2. It can be seen that for this value of a (i.e., $a = 0.25$), the log shrinkage function is quite similar to the the nn-garrote shrinkage function, but it attenuates large y slightly more than the nn-garrote shrinkage function.

4 Vector Case

Consider the vector case,

$$F(\mathbf{x}) = \frac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \sum_n \phi(x(n)) \tag{40}$$

where

$$\mathbf{x} = [x(1), \ldots, x(N)]^T$$

$$\mathbf{y} = [y(1), \ldots, x(M)]^T$$

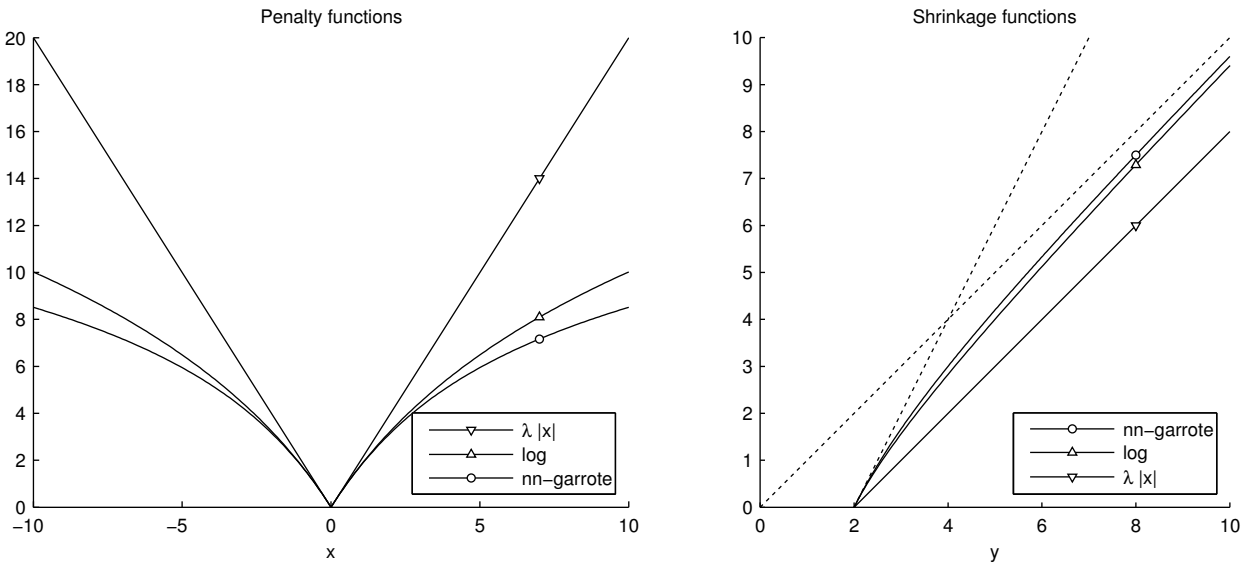

Figure 11: Comparison of penalty and shrinkage functions.

and $\mathbf{A}$ is a matrix of size $M \times N$.

The function $F(\mathbf{x})$ can not be easily minimized unless the penalty function $\phi(x)$ is quadratic. When $\phi(x)$ is quadratic, then the derivative of $F(\mathbf{x})$ with respect to $\mathbf{x}$ gives linear equations, and the minimizer $\mathbf{x}$ can be obtained by solving a linear system. However, when $\phi(x)$ is quadratic, the solution $\mathbf{x}$ will generally not be sparse; hence other penalty functions are preferred for sparse signal processing, as described in the preceding sections.

To minimize $F(\mathbf{x})$, an iterative numerical algorithm must be used. One approach to derive suitable algorithms is based on the majorization-minimization (MM) method from optimization theory [7]. Instead of minimizing $F(\mathbf{x})$ directly, the MM method solves a sequence of simpler minimization problems

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} G_k(\mathbf{x}) \tag{41}$$

where k is the iteration counter, $k = 1, 2, 3, \ldots$. With initialization $\mathbf{x}_0$, the expression in (41) is a rule for updating $\mathbf{x}_k$. The MM method asks that each function $G_k(\mathbf{x})$ be a majorizer (upper bound) of $F(\mathbf{x})$ and that it coincides with $F(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_k$. That is,

$$G_k(\mathbf{x}) \geq F(\mathbf{x}) \quad \text{for all } \mathbf{x} \tag{42}$$

$$G_k(\mathbf{x}_k) = F(\mathbf{x}_k) \tag{43}$$

To apply the MM method to minimize $F(\mathbf{x})$ in (40), one may either majorize the data fidelity term $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ (as in ISTA [6,8], FISTA [2], etc), or the penalty term $\sum_n \phi(x(n))$, or both terms. In the following, we majorize just the penalty term.

To majorize the penalty term, it will be useful to consider first the scalar case. Specifically, we first find a majorizer for $\phi(x)$ with $x \in \mathbb{R}$. The majorizer $g(x)$ should be an upper bound for $\phi(x)$

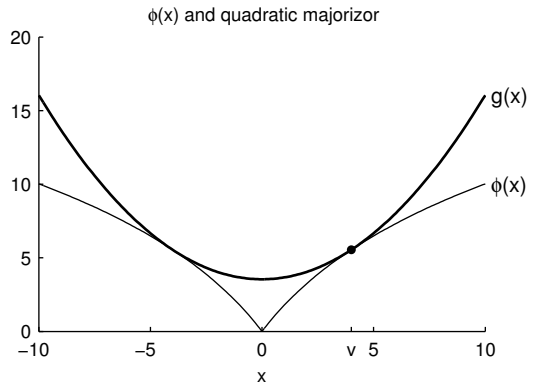

Figure 12: The penalty function and its quadratic majorizer.

that coincides with $\phi(x)$ at a specified point $v \in \mathbb{R}$ as shown in Fig. 12. That is,

$$g(x) \geq \phi(x) \quad \text{for all } x \in \mathbb{R} \tag{44}$$

$$g(v) = \phi(v) \quad \text{where } v \text{ is a specified point} \tag{45}$$

We will find a quadratic majorizer $g(x)$ of the form

$$g(x) = mx^2 + c. \tag{46}$$

For this quadratic majorizer, the conditions (44) and (45) are equivalent to:

$$g(v) = \phi(v) \tag{47}$$

$$g'(v) = \phi'(v). \tag{48}$$

That is, $g(x)$ and its derivative should agree with $\phi(x)$ at $x = v$. Using (46), these two conditions can be written as

$$mv^2 + b = \phi(v) \tag{49}$$

$$2mv = \phi'(v). \tag{50}$$

Solving for m and b gives

$$m = \frac{\phi'(v)}{2v}, \qquad b = \phi(v) - \frac{v}{2}\phi'(v).$$

The majorizer $g(x)$ in (46) is therefore given by

$$g(x) = \frac{\phi'(v)}{2v}\, x^2 + \phi(v) - \frac{v}{2}\phi'(v). \tag{51}$$

With this function $g(x)$ we have $g(x) \geq \phi(x)$ with equality at $x = v$ as illustrated in Fig. 12. To emphasize the dependence of $g(x)$ on the point v, we write

$$g(x, v) = \frac{\phi'(v)}{2v}\, x^2 + \phi(v) - \frac{v}{2}\phi'(v). \tag{52}$$

The scalar majorizer can be used to obtain a majorizer for $F(\mathbf{x})$ in (40). If $\mathbf{x}$ and $\mathbf{v}$ denote vectors

$$\mathbf{x} = [x(1), \ldots, x(N)]^T$$
$$\mathbf{v} = [v(1), \ldots, v(N)]^T$$

then

$$\sum_n g(x(n), v(n)) \geq \sum_n \phi(x(n)) \tag{53}$$

with equality if $\mathbf{x} = \mathbf{v}$. That is, the left-hand-side of (53) is a majorizer for $\sum_n \phi(x(n))$.

Note that the left-hand-side of (53) can be written compactly as

$$\sum_n g(x(n), v(n)) = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c \tag{54}$$

where

$$\mathbf{W} = \begin{bmatrix} \dfrac{\phi'(v(1))}{v(1)} & & \\ & \ddots & \\ & & \dfrac{\phi'(v(N))}{v(N)} \end{bmatrix} \tag{55}$$

and

$$c = \sum_n \phi(v(n)) - \frac{v}{2}\phi'(v(n)). \tag{56}$$

As an abbreviation, we denote the diagonal matrix $\mathbf{W}$ as

$$\mathbf{W} = \text{diag}(\phi'(\mathbf{v})./\mathbf{v}) \tag{57}$$

where the notation './' denotes component-wise division.

Therefore, using (53), a majorizer for $F(\mathbf{x})$ is given by

$$G(\mathbf{x}, \mathbf{v}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c, \quad \text{where } \mathbf{W} = \text{diag}(\phi'(\mathbf{v})./\mathbf{v}). \tag{58}$$

Minimizing $G(\mathbf{x}, \mathbf{v})$ with respect to $\mathbf{x}$ gives

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \mathbf{W})^{-1}\mathbf{A}^T\mathbf{y} \tag{59}$$

where $\mathbf{W}$ depends on $\mathbf{v}$.

Therefore, the MM update in (41) with $G_k(\mathbf{x}) = G(\mathbf{x}, \mathbf{x}_k)$ produces the sequence

$$\mathbf{x}_{k+1} = (\mathbf{A}^T\mathbf{A} + \mathbf{W})^{-1}\mathbf{A}^T\mathbf{y} \quad \text{where } \mathbf{W} = \text{diag}(\phi'(\mathbf{x}_k)./\mathbf{x}_k). \tag{60}$$

Note that if components of $\mathbf{x}$ go to zero, the entries of $\mathbf{W}$ go to infinity. Hence, the update equation (60) may become numerically inaccurate. Moreover, because the solution $\mathbf{x}$ is expected to be sparse, components of $\mathbf{x}_k$ will go to zero as the iteration progresses. This problem is avoided, as described in Ref. [7], by using the matrix inverse lemma to write

$$(\mathbf{A}^T\mathbf{A} + \mathbf{W})^{-1} = \mathbf{W}^{-1} - \mathbf{W}^{-1}\mathbf{A}^T(\mathbf{I} + \mathbf{A}\mathbf{W}^{-1}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{W}^{-1}. \tag{61}$$

Table 1: Quadratic MM algorithm for minimizing (4).

Input: $\mathbf{y} \in \mathbb{R}^M$, $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\phi(\cdot)$
Output: $\mathbf{x} \in \mathbb{R}^N$

1: $\mathbf{x} = \mathbf{y}$
2: $\mathbf{g} = \mathbf{A}^T\mathbf{y}$
3: **repeat**
4: $\quad \mathbf{\Lambda}(n,n) \leftarrow x(n)/\phi'(x(n)), \quad n = 1, \ldots, N$
5: $\quad \mathbf{F} = \mathbf{I} + \mathbf{A}\mathbf{\Lambda}\mathbf{A}^T$
6: $\quad \mathbf{x} = \mathbf{\Lambda}(\mathbf{g} - \mathbf{A}^T\mathbf{F}^{-1}\mathbf{A}\mathbf{\Lambda}\mathbf{g})$
7: **until** convergence
8: **return** $\mathbf{x}$

Note, as components of $\mathbf{x}_k$ go to zero, the entries of $\mathbf{W}^{-1}$ go to zero instead of to infinity.

For convenience, we define

$$\mathbf{\Lambda} := \mathbf{W}^{-1} = \text{diag}(\mathbf{x}_k./\phi'(\mathbf{x}_k)). \tag{62}$$

Using (61), the quadratic MM update in (60) is given by

$$\mathbf{x}_{k+1} = \mathbf{\Lambda}\mathbf{A}^T\mathbf{y} - \mathbf{\Lambda}\mathbf{A}^T(\mathbf{I} + \mathbf{A}\mathbf{\Lambda}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{\Lambda}\mathbf{A}^T\mathbf{y}. \tag{63}$$

An implementation of the quadratic MM algorithm based on this update equation is given in Table 1. A MATLAB program for this algorithm is given in Sec. 8. This iterative algorithm may be considered a form of the FOCUSS algorithm [13].

Note that if $\mathbf{A}$ is banded, then the matrix $\mathbf{I} + \mathbf{A}\mathbf{\Lambda}\mathbf{A}^T$ is banded and fast algorithms for banded systems can be used to implement the algorithm.

Note that the penalty function $\phi(\cdot)$ arises in the update equations only as entries of $\mathbf{\Lambda}$ in the expression $x/\phi'(x)$. Hence, it is informative to analyze and simplify the function $x/\phi'(x)$.

Example 1: The function $x/\phi'(x)$ is given by

$$\frac{x}{\phi_1'(x)} = \frac{x}{\lambda\,\text{sign}(x)} \tag{64}$$

or more simply as

$$\frac{x}{\phi_1'(x)} = \frac{1}{\lambda}|x| \tag{65}$$

Notice that using (64) leads to a numerical issue when $x = 0$ because the function $\text{sign}(x)$ is undefined at $x = 0$. On the other hand, the simplified form in (65) is well defined for all x.

Example 2:

$$\frac{x}{\phi_2'(x)} = \frac{x}{\frac{\lambda}{1+a|x|}\,\text{sign}(x)} \tag{66}$$

or more simply as

$$\frac{x}{\phi_2'(x)} = \frac{1}{\lambda}\,|x|\,(1 + a|x|). \tag{67}$$

For large x, we have $x/\phi_2'(x) \approx ax^2/\lambda$.

Example 3:

$$\frac{x}{\phi_3'(x)} = \frac{1}{2\lambda^2}\,|x|\left(\sqrt{x^2 + 4\lambda^2} + |x|\right). \tag{68}$$

As for Example 2, for large x, we have $x/\phi_3'(x) \approx x^2/\lambda^2$.

For these three penalty functions, it can be seen that if $x_k(m)$ becomes equal to zero at some iteration k for some $1 \leq m \leq N$, then it will remain zero for all the remaining iterations. This is because $x(m)/\phi'(x(m))$ will equal zero. This would appear to interfere with convergence of the algorithm to the minimizer of (4). However, detailed investigation in [12] suggests that this phenomenon does not seem to inhibit the reliable operation of this iterative algorithm.

5 Optimality Conditions

When the penalty function $\phi(x)$ is convex, then the minimizer of the cost function (4) must satisfy specific conditions [1, Prop 1.3]. These conditions can be used to verify the optimality of a solution produced by a numerical algorithm. When the penalty function $\phi(x)$ is not convex, then conditions can be used to determine if a local minimum is attained.

If $\phi(x)$ is convex and differentiable, and if $\mathbf{x}$ minimizes (4), then $\mathbf{x}$ must satisfy:

$$[\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n = \phi'(x(n)), \quad n = 1, \ldots, N \tag{69}$$

where $[\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n$ denotes the n-th component of the vector $\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})$. However, this can not be applied directly when $\phi'(x)$ is not differentiable as in the three example penalty functions in the preceding sections.

If $\phi(x)$ is convex but not differentiable, then the optimality condition is given by

$$[\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n \in \partial\phi(x(n)), \quad n = 1, \ldots, N \tag{70}$$

where $\partial\phi(\cdot)$ is the subdifferential, a *set*-valued generalization of the derivative [1, Prop 1.3]. For Example 1, with $\phi_1(x) = \lambda|x|$, the subdifferential is given by

$$\partial\phi_1(x) = \begin{cases} \{\lambda\}, & x > 0 \\ [-\lambda, \lambda], & x = 0 \\ \{-\lambda\}, & x < 0. \end{cases} \tag{71}$$

In this case, the condition (70) is given by

$$\begin{cases} [\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n = \lambda, & x(n) > 0 \\ -\lambda \leq [\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n \leq \lambda, & x(n) = 0 \\ [\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})]_n = -\lambda, & x(n) < 0 \end{cases} \tag{72}$$

as illustrated in Fig. 14 below.

5.1 Setting the threshold

The condition (70) can sometimes be used as a guideline to set the threshold value for the penalty function $\phi(x)$. Suppose the data $y(n)$ is modeled as $\mathbf{Ax}$ where $\mathbf{x}$ is sparse and $\mathbf{w}$ is additive noise,

$$\mathbf{y} = \mathbf{Ax} + \mathbf{w}. \tag{73}$$

Note that if $\mathbf{x} = \mathbf{0}$, then $\mathbf{y}$ consists of noise only (i.e. $\mathbf{y} = \mathbf{w}$). Then (70) gives

$$[\mathbf{A}^T\mathbf{w}]_n \in \partial\phi(0). \tag{74}$$

For $\phi_1(x) = \lambda|x|$, that is

$$-\lambda \leq [\mathbf{A}^T\mathbf{w}]_n \leq \lambda, \quad n = 1, \ldots, N. \tag{75}$$

This implies that λ should be chosen so that $\lambda \geq \max|\mathbf{A}^T\mathbf{w}|$ where $\mathbf{w}$ is the additive noise. On the other hand, the larger λ is, the more the signal $\mathbf{x}$ will be attenuated. (If λ is too large, then the solution to (4) will be $\mathbf{x} = \mathbf{0}$.) Hence, it is reasonable to set λ to the smallest value satisfying (75), namely,

$$\lambda \approx \max|\mathbf{A}^T\mathbf{w}|. \tag{76}$$

Although (76) assumes availability of the noise signal $\mathbf{w}$, which is not realistic in practice, (76) can often be estimated based on knowledge of statistics of the noise $\mathbf{w}$. For example, if the noise $\mathbf{w}$ is white Gaussian, then the '3σ' rule can be used to estimate the maximum value of $|\mathbf{A}^T\mathbf{w}|$.

6 Deconvolution Example

The deconvolution problem is one of finding the input signal $x(n)$ to a linear time-invariant system when the output signal $y(n)$ and the impulse response of the system $h(n)$ are both known. The problem is sometimes ill conditioned, depending on $h(n)$; and the output signal is usually noisy. When it is known that the input signal is sparse, then a suitable approach for estimating the input signal is to solve the optimization problem (4) where the penalty function is chosen so as to promote sparsity, for example, any of the three penalty functions in the preceding sections.

The noisy output signal $y(n)$ is given by

$$y(n) = (h * x)(n) + w(n), \qquad w \sim \mathcal{N}(0, \sigma^2) \tag{77}$$

where $*$ denotes convolution. This can be written as

$$\mathbf{y} = \mathbf{Hx} + \mathbf{w} \tag{78}$$

where $\mathbf{H}$ is the appropriately defined convolution matrix. If the impulse response $h(n)$ is finite in length, then the matrix $\mathbf{H}$ will be banded. Most of the entries of a banded matrix are zero, so it can be stored with less memory; also, linear algebra operations, like matrix-vector multiplication and Gaussian elimination for solving linear systems, are fast for banded matrices compared with full (dense) matrices.

A solution to the sparse deconvolution can be obtained using a specified penalty function $\phi(x)$,

$$\mathbf{x}_\phi = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \sum_n \phi(x(n)). \tag{79}$$

In the proceeding examples, $\phi(x)$ has a parameter λ that determines the threshold of the shrinkage function.

How should λ be set? Combining (76) with the '3-sigma' rule, gives

$$\lambda \approx \max|\mathbf{H}^T\mathbf{w}| \approx 3\,\mathrm{std}(\mathbf{H}^T\mathbf{w}) \tag{80}$$

where std() denotes the standard deviation. If the noise $w(n)$ is zero-mean white Gaussian with variance σ^2, then the standard deviation of $\mathbf{H}^T\mathbf{w}$ is given by

$$\mathrm{std}(\mathbf{H}^T\mathbf{w}) = \sigma\sqrt{\sum_n |h(n)|^2} = \sigma\|\mathbf{h}\|_2. \tag{81}$$

Hence, we get a 'rule of thumb':

$$\lambda \approx 3\,\sigma\sqrt{\sum_n |h(n)|^2} = 3\,\sigma\|\mathbf{h}\|_2. \tag{82}$$

Figure 13a shows a sparse signal $x(n)$. This signal $x(n)$ is convolved with the 4-point impulse response

$$h(n) = \begin{cases} \frac{1}{4} & n = 0, 1, 2, 3 \\ 0 & \text{otherwise} \end{cases} \tag{83}$$

and white Gaussian noise $w(n)$ is added to obtain the observed signal $y(n)$ illustrated in Fig. 13b. The impulse response has $\|\mathbf{h}\|_2 = 0.5$. The standard deviation of the AWGN is $\sigma = 0.03$. Using (82), we set $\lambda = 0.045$. Using the penalty functions $\phi_1(x) = \lambda|x|$ (i.e., ℓ_1 norm solution) and $\phi_3(x)$ in (38) (i.e., nn-garrote solution), and using the quadratic MM algorithm, we obtain the two sparse solutions shown in Fig.13c,d. Both recover the sparse input signal quite accurately.

It is interesting to compare the sparse solution with the least square solution,

$$\mathbf{x}_{\mathrm{LS}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 \tag{84}$$

which is shown in Fig.13e. Note that the least square solution is unable to recover the sparse property of the input signal, and unable to suppress the noise as effectively as the sparse solution.

The optimality of the two sparse solutions are verified in Fig. 14. For the nn-garrote solution, which corresponds to a non-convex penalty function, the solution might not be a global minimizer of (79), so it is desirable to verify that the solution is a local minimum. The graphs in Fig. 14 show $[\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{x})]_n$ vs $x(n)$ as scatter plots for each of the two sparse solutions. Overlaid on the graph is the function $\phi'(x)$. In each case, the points of the scatter plot coincide with $\phi'(x)$, which for a convex penalty function indicates the optimality of the solution.

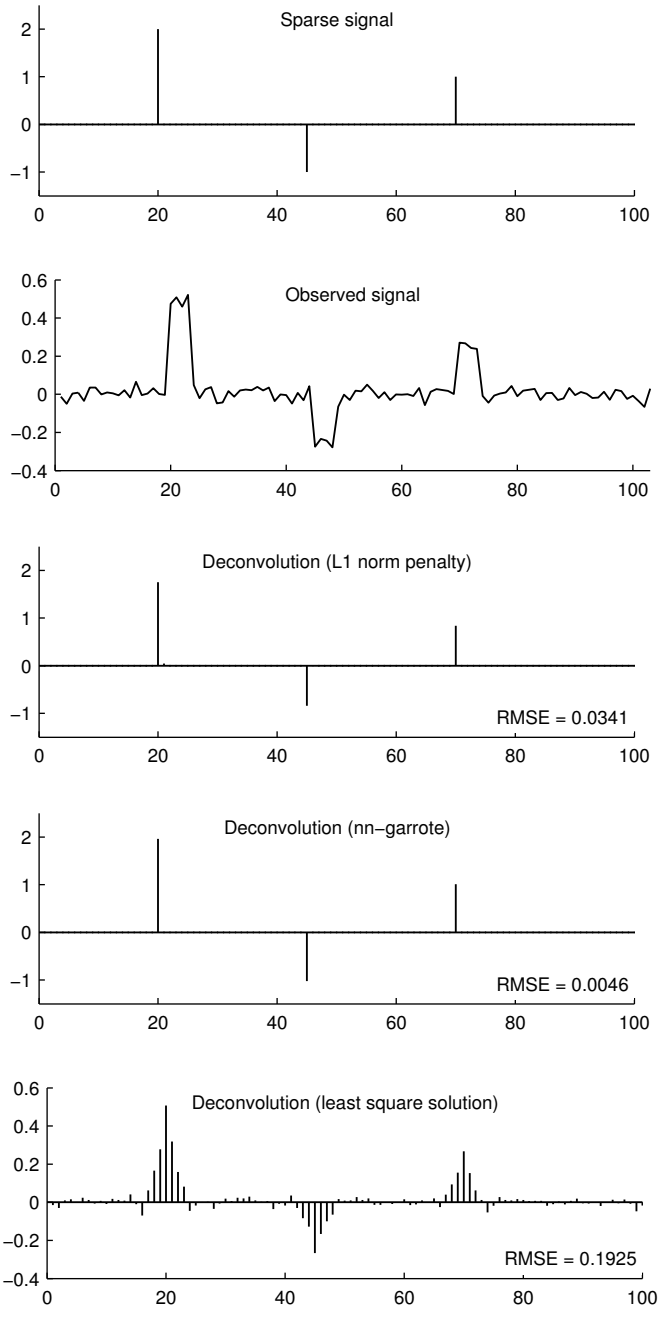

Figure 13: Deconvolution example.

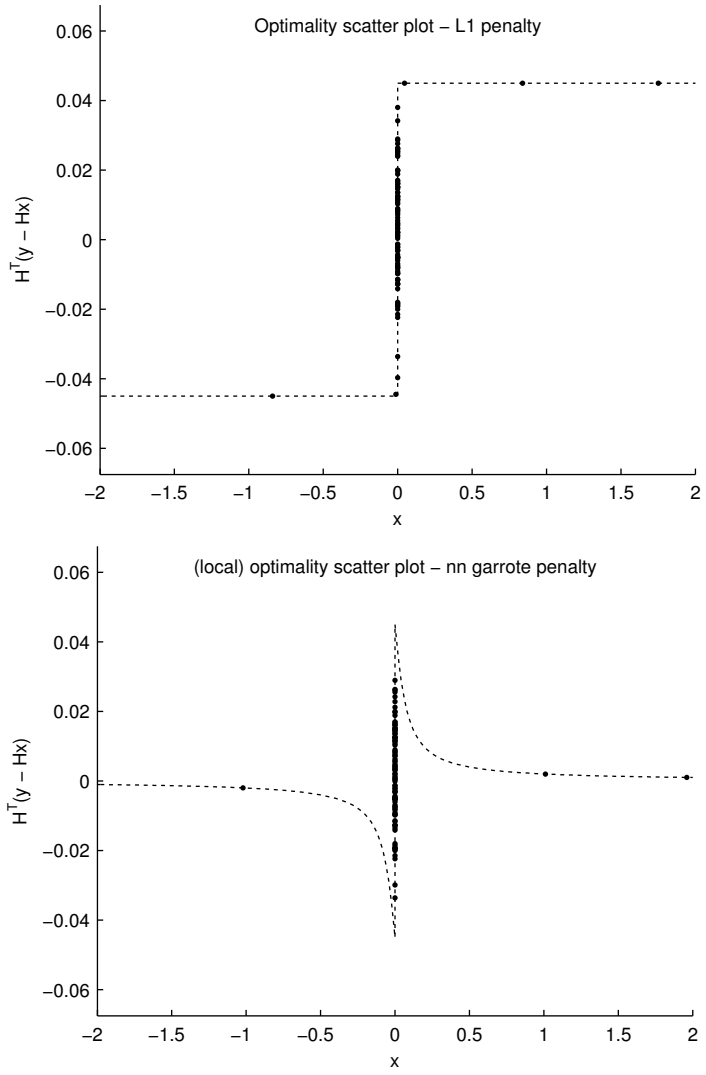

Figure 14: Optimality conditions.

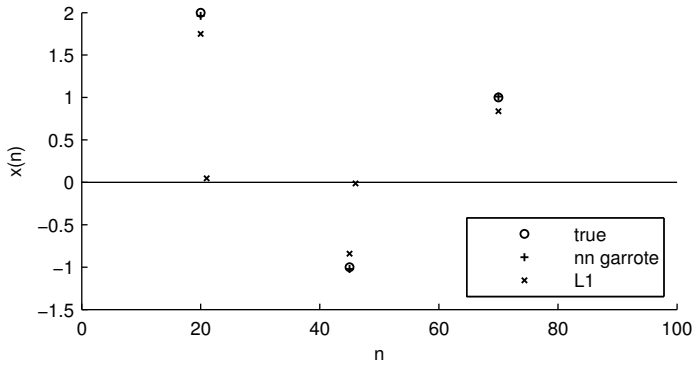

Figure 15: Comparison of ℓ_1 norm and nn-garrote penalties.

To compare the ℓ_1 norm and nn-garrote solutions in more detail, Fig. 15 shows both solutions on the same axis. The original sparse signal is also shown. For clarity, only the non-zero values of each signal are displayed. It can be seen that the ℓ_1 norm solution is slightly attenuated compared with the nn-garrote solution. This is to be expected — recall that the shrinkage function corresponding to the ℓ_1 norm penalty function is the soft thresholding function which subtracts λ from every value greater than the threshold. Hence this penalty function has the effect of attenuating the large signal values. In contrast, the nn-garrote penalty function attenuates large values much less. The difference in these two solutions is due to the different behavior of the two penalty functions $\phi_1(x)$ and $\phi_3(x)$. The nn-garrote penalty function $\phi_3(x)$ increases more slowly than the ℓ_1-norm penalty function $\phi_1(x)$ (see Fig. 11).

If further adjustment of the penalty function is desired, one may use the log penalty function $\phi_2(x)$ in (6) and adjust the parameter a, or use other penalty functions in the literature.

A MATLAB program for implementing the quadratic MM algorithm for sparse deconvolution is given in Sec. 8.

7 Exercises

1. For the cost function
$$\phi(x) = \lambda |x|^p \tag{85}$$
where $0 < p < 1$, conduct the derivation as in the notes. That is, find the shrinkage function, find its derivative at the threshold, etc. Plot the shrinkage function, and other functions, for a specific value of p, e.g. $p = 0.7$. Is the function $x + \phi'(x)$ monotonic? Do any complications arise in the derivation of the shrinkage function?

2. For the cost function
$$\phi(x) = \frac{a|x|}{|x| + b} \tag{86}$$
conduct the derivation as in the notes: Find the shrinkage function and its threshold, etc.

8 MATLAB Programs

Program for sparse deconvolution using quadratic MM.

```
function [x, cost] = deconv_MM(y, phi, wfun, H, Nit)
% [x, cost] = deconv_MM(y, phi, wfun, H, Nit)
% Sparse deconvolution
% Cost function : 0.5 * sum(abs(y-H(x)).^2) + sum(phi(x));
%
% INPUT
%   y - noisy data
%   H - banded convolution matrix [sparse data structure]
%   phi - cost function
%   wfun - function x / phi'(x) - needed for quadratic MM
%   Nit - number of iterations
%
% OUTPUT
%   x - deconvolved sparse signal
%   cost - cost function history

% Reference: Penalty and Shrinkage Functions for Sparse Signal Processing
% Ivan Selesnick, selesi@poly.edu, 2012
% http://eeweb.poly.edu/iselesni/lecture_notes/

% Algorithm: majorization-minimization with banded system solver.

y = y(:);                                           % convert to column vector
cost = zeros(1, Nit);                               % cost function history
[M, N] = size(H);
x = y;                                              % initialization
g = H'*y;                                           % H'*y
for k = 1:Nit
    Lam = spdiags(wfun(x), 0, N, N);                % Lam : diagonal matrix
    F = speye(M) + H*Lam*H';                        % F : banded matrix
    x = Lam * (g - (H'*(F\(H*(Lam*g)))));           % update x (solve banded system)
    cost(k) = 0.5*sum(abs(y-H*x).^2) + sum(phi(x)); % cost function value
end
```

Sparse deconvolution example using nn-garrote penalty.

```
%% Sparse deconvolution example
% using the nn-garrote penalty function

%% Create data

clear
N = 100;                    % N : length of signal
s = zeros(N,1);
k = [20 45 70];             % k : spike positions
a = [2 -1 1];               % a : spike amplitudes
```

```
s(k) = a;                   % s : spike signal

h = [1 1 1 1]/4;            % h : impulse response
M = N + 3;                  % M : length of observed signal
sigma = 0.03;               % sigma : standard deviation of AWGN
randn('state',0);           % set state so that example can be reproduced
w = sigma * randn(M,1);     % w : additive zero-mean Gaussian noise (AWGN)
y = conv(h,s) + w;          % y : observed data

figure(1), clf
subplot(2,1,1)
plot(y)
xlim([0 M])
title('Observed signal');

%% Create convolution matrix H

H = sparse(M,N);                            % sparse matrix structure
e = ones(N,1);
for i = 0:3
    H = H + spdiags(h(i+1)*e, -i, M, N);    % H : convolution matrix (sparse)
end
issparse(H)                                 % confirm that H is a sparse matrix

err = H*s - conv(h,s);                      % Verify that H*s is the same as conv(h,s)
fprintf('Maximum error = %g\n', max(abs(err)))

%% Sparse deconvolution (nn garrote)
% Run the quadratic MM algorithm

lam = 3 * sigma * sqrt(sum(abs(h).^2));   % lam : threshold using '3-sigma' rule

dphi = @(x) 0.5*(-abs(x) + sqrt( x.^2 + 4*lam^2)).*sign(x);
phi = @(x) lam^2*asinh(abs(x)/(2*lam)) + lam^2 * abs(x)./(sqrt(4*lam^2+x.^2) + abs(x));
wfun = @(x) 1/(2*lam^2) * abs(x).*(sqrt(x.^2+4*lam^2) + abs(x));  % x / dphi(x)

Nit = 50;                               % Nit : number of iterations
[x, cost] = deconv_MM(y, phi, wfun, H, Nit);

figure(1)
stem(x, 'marker', 'none')
ylim([-1.5 2.5]);
title('Deconvolution (nn-garrote)');

%% Display cost function history

figure(2)
plot(1:Nit, cost, '.-')
title('Cost function history');
xlabel('Iteration')
xlim([0 Nit])

%% Verify (local) optimality conditions
```

```
v = H'*(y-H*x);
t = [linspace(-2, -eps, 100) linspace(eps, 2, 100)];

figure(2)
plot(x, v, '.', 'markersize', 6)
line(t, dphi(t), 'linestyle', ':')
ylim([-1 1]*lam*1.5)
xlim(t([1 end]))
xlabel('x')
ylabel('H^T(y - Hx)')
title('(local) optimality scatter plot - nn garrote penalty');
```

References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2(1):183–202, 2009.

[3] L. Breiman. Better subset selection using the nonnegative garrote. *Technometr.*, 37(4):373–384, 1995.

[4] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. Image Process.*, 6(2):298–311, February 1997.

[5] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer-Verlag (New York), 2010.

[6] I. Daubechies, M. Defriese, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math*, LVII:1413–1457, 2004.

[7] M. Figueiredo, J. Bioucas-Dias, and R. Nowak. Majorization-minimization algorithms for wavelet-based image restoration. *IEEE Trans. Image Process.*, 16(12):2980–2991, December 2007.

[8] M. Figueiredo and R. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Trans. Image Process.*, 12(8):906–916, August 2003.

[9] M. A. T. Figueiredo and R. D. Nowak. Wavelet-based image estimation: An empirical Bayes approach using Jeffrey's noninformative prior. *IEEE Trans. Image Process.*, 10(9):1322–1331, September 2001.

[10] H. Gao. Wavelet shrinkage denoising using the nonnegative garrote. *J. Comput. Graph. Statist.*, 7:469–488, 1998.

[11] A. Hyvärinen. Sparse code shrinkage: Denoising of non-Gaussian data by maximum likelihood estimation. *Neural Computation*, 11:1739–1768, 1999.

[12] J. Oliveira, J. Bioucas-Dias, and M. A. T. Figueiredo. Adaptive total variation image deblurring: A majorization-minimization approach. *Signal Processing*, 89(9):1683–1693, September 2009.

[13] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado. Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Process.*, 51(3):760–770, March 2003.

[14] B. Vidakovic. *Statistical Modeling by Wavelets*. John Wiley & Sons, 1999.