Compressed Sensing and Sparse Signal Processing

Wu-Sheng Lu

University of Victoria, Canada

November 2010

List of Contents

I. Digital Signals: Sampling and Representation

1.1 The Shannon-Nyquist Sampling Theorem

The Shannon-Nyquist Sampling Theorem:
A continuous-time signal $x(t)$ with frequencies no higher than f_{max} (in Hz) *can be reconstructed from its samples $x[k] = x(kT_s)$, if the samples are taken at a rate $f_s = 1/T_s$ that is greater than 2 f_{max}.*

The frequency 2 f_{max} is called the *Nyquist rate*. Obviously, the above theorem can be stated as:
A continuous-time signal $x(t)$ with frequencies no higher than f_{max} (in Hz) *can be reconstructed from its samples $x[k] = x(kT_s)$, if the samples are taken at the Nyquist rate.*

There are several ways to explain this important theorem. In what follows, we present a graphics-based explanation and an analytic explanation of the theorem.

A. A Graphics-Based Explanation

The explanation here is given in the frequency domain by examining the spectrum of the sampled signal in relation to that of the original continuous-time signal.
An analog signal is said to be *band-limited*, if it has an identifiable maximum frequency in its spectrum, say, f_{max} Hz. Obviously, the signal $x(t)$ in Shannon's theorem is band-limited.
There are great many real-world signals that are band-limited. For example, speech and music are always band-limited. To process signals that are not band-limited, it is often of convenience to deal with their band-limited counterparts by lowpass or bandpass filtering the signals as a pre-processing step. This step is often an integral part of a DSP system. In Fig. 1.1, this step is the first function block, and the filter that turns the input signal into a band-limited signal is called *anti-aliasing* filter.
Without the loss of generality, let us now consider a band-limited continuous-time signal $x(t)$ whose spectrum is within the region $0 \le \Omega \le \Omega_{max}$ where $\Omega_{max} = 2\pi f_{max}$. Suppose signal $x(t)$ is defined for $-\infty < t < \infty$ and is sampled uniformly at $t = nT_s$, where $T_s = 1/f_s$ denotes the sampling period in seconds (this means that f_s is the sampling frequency) and $-\infty < n < \infty$ are integers. The diagram in Fig. 2.4 describes this ideal sampling model.

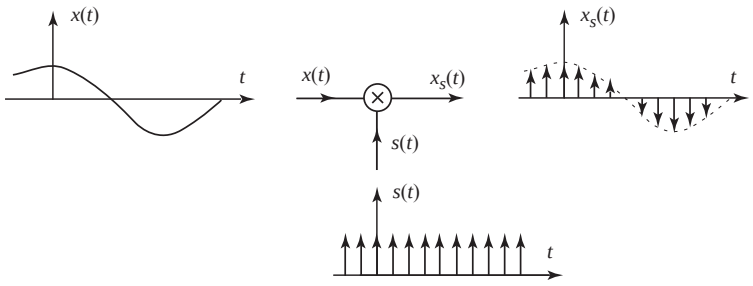


Figure 1.1

In Fig. 1.1, $s(t)$ is the periodic impulse train

$$s(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s)$$

where $\delta(t)$ is the unit impulse function, and the sampled signal is obtained by modulating $s(t)$ with $x(t)$ as

$$x_s(t) = x(t)s(t) = x(t)\sum_{k=-\infty}^{\infty}\delta(t-kT_s) \tag{1.1}$$

which gives

$$x_s(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\delta(t-kT_s) \tag{1.2}$$

The sampling process is described in Fig. 1.1 and can be analyzed in the *frequency domain* as follows. Applying Fourier transform to (1.1) and note that the Fourier transform of a product of two functions is equal to the convolution of the Fourier transforms of these functions, i.e.,

$$X_s(j\Omega) = \frac{1}{2\pi}X(j\Omega) * S(j\Omega)$$

By applying the Fourier-series kernel theorem, we obtain

$$S(j\Omega) = \frac{2\pi}{T_s}\sum_{k=-\infty}^{\infty}\delta(\Omega - k\Omega_s)$$

where $\Omega_s = 2\pi/T_s$ is the angular sampling frequency in radians/sec. Therefore,

$$X_s(j\Omega) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X(j(\Omega - k\Omega_s)) \tag{1.3}$$

Equation (1.3) is important because it relates explicitly the spectrum of the sampled analog signal $x_s(t)$ to that of the original analog signal $x(t)$. We see that the Fourier transform of $x_s(t)$ consists of periodically repeated copies of the Fourier transform of $x(t)$. More specifically, Eq. (1.3) says that the copies of $X(j\Omega)$ are *shifted* by integer multiples of the sampling frequency and then *superimposed* to generate the periodic Fourier transform of the impulse train of samples. Two representative cases in terms of the value of $\Omega_{\max}$ compared with that of $\Omega_s - \Omega_{\max}$ are shown in Figs. 1.2 and 1.4.

Case I: The sampling frequency is sufficiently high such that

$$\Omega_s - \Omega_{\max} > \Omega_{\max} \qquad \text{i.e.,} \qquad \Omega_s > 2\Omega_{\max} \tag{1.4}$$

In this case, the replicas of $X(j\Omega)$ do not overlap, see Fig. 1.2c. Consequently, the continuous-time signal $x(t)$ can be recovered by ideal lowpass filtering as illustrated in Fig. 1.3.

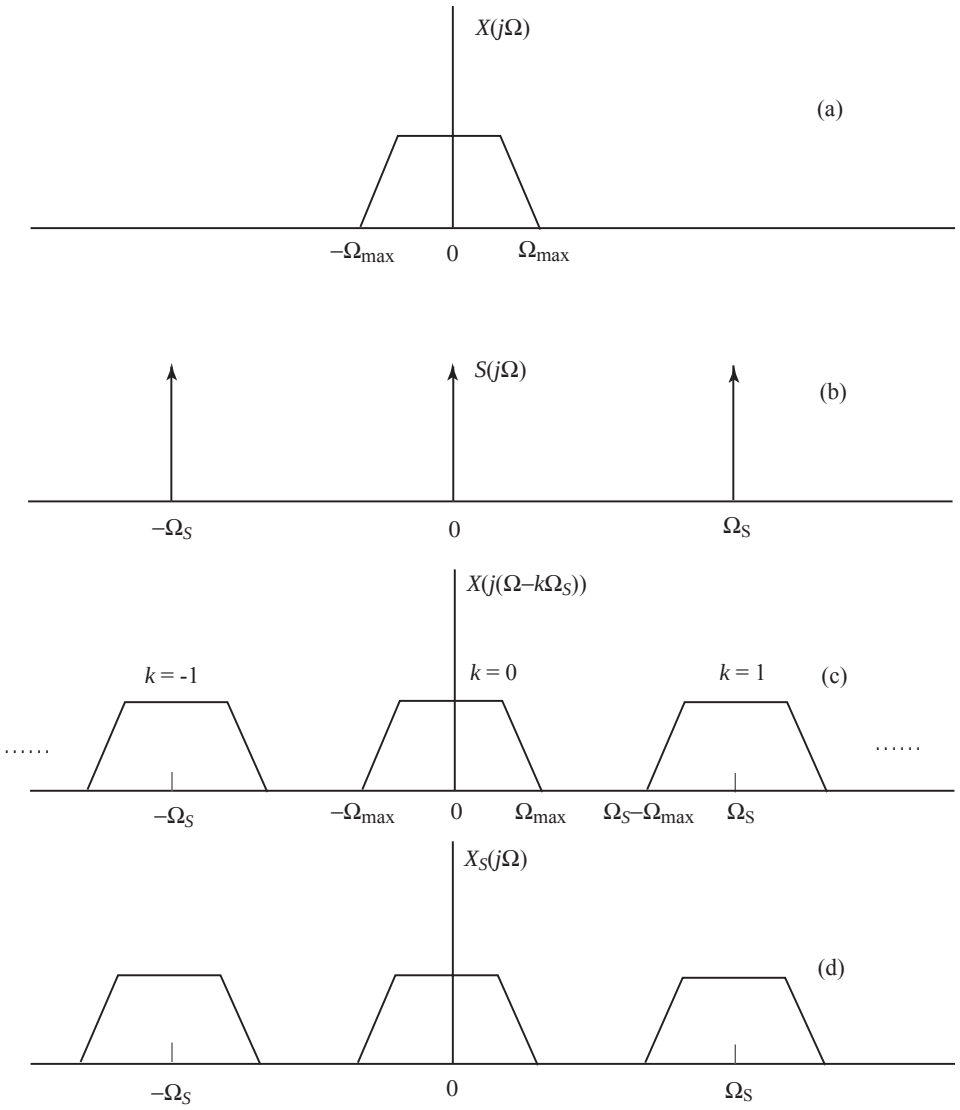

X(jΩ)
(a)
−Ωmax
0
Ωmax
S(jΩ)
(b)
−ΩS
0
ΩS
X(j(Ω−kΩS))
k = -1
k = 0
k = 1
(c)
.....
.....
−ΩS
−Ωmax
0
Ωmax
ΩS−Ωmax
ΩS
XS(jΩ)
(d)
−ΩS
0
ΩS

Figure 1.2

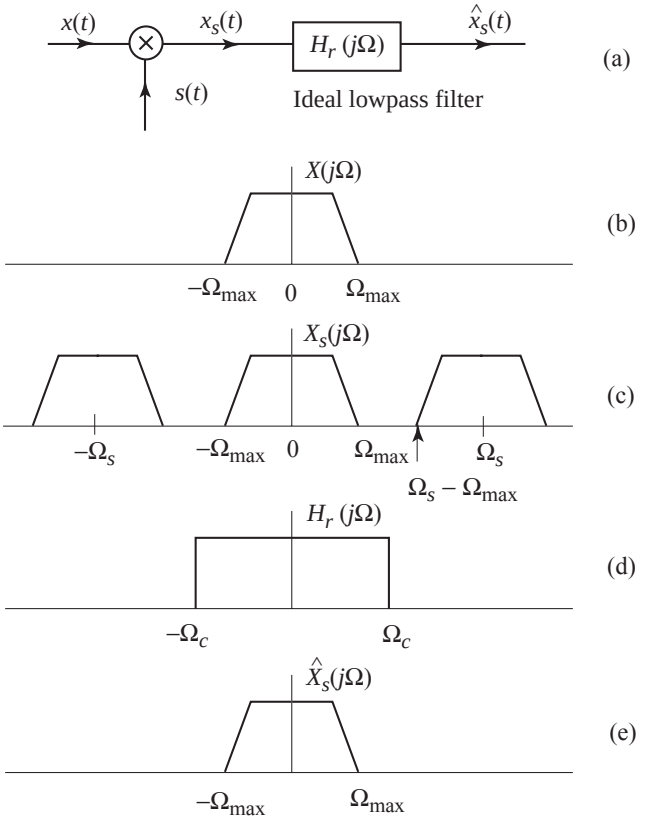


Figure 1.3

From Fig. 1.3d, we see that the ideal lowpass filter should have a cutoff frequency Ω_c satisfying

$$\Omega_{\max} \le \Omega_c \le \Omega_s - \Omega_{\max} \tag{1.5}$$

and it can be readily verified that the choice $\Omega_c = \Omega_s / 2$ does meet the constraints in (1.5).

Case II: The sampling frequency is not high enough, namely,

$$\Omega_s - \Omega_{\max} < \Omega_{\max} \qquad \text{i.e.,} \qquad \Omega_s < 2\Omega_{\max}$$

In this case the copies of $X(j\Omega)$ overlaps with each other (see Fig. 1.4c) and, when they are added up (see Eq. (1.3)), $X(j\Omega)$ is no longer recoverable by lowpass filtering of the sampled signal $x_s(t)$.

From the analysis above, we conclude that a band-limited $x(t)$ can be recovered by its samples if condition (1.4) holds. Note that the condition in (1.4) is equivalent to

$$f_s > 2 f_{\max} \tag{1.6}$$

which is exactly the condition stated in Shannon's theorem. In the literature, the condition in (1.4) (and equivalently in (1.6)) is sometimes referred to as the *Nyquist condition*.

Examples

(a) In digital telephony, a 3.4 KHz signal bandwidth is considered acceptable for telephone conversations. According to Shannon's sampling theorem, a sampling rate of 8 KHz which is greater

than two times of the acceptable signal bandwidth is obviously appropriate.
(b) In analog music signal processing, a bandwidth of 20 KHz preserves the fidelity. Current compact disc (CD) music systems adopt a sampling rate of 44.1 KHz which is more than twice of the 20 KHz bandwidth and is therefore quite adequate.

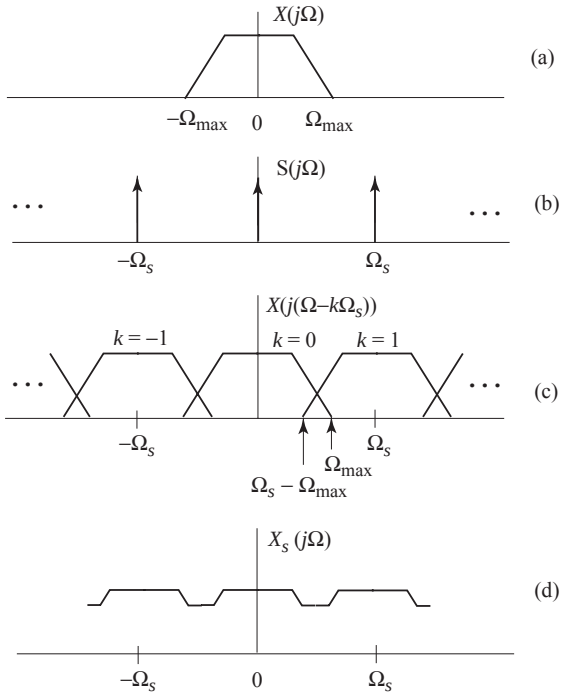


Figure 1.4

B. <u>Recovery of the Continuous-Time Signal: An Analytical Formula</u>

We have seen that if the discrete-time sequence $\{x[k] = x(kT_s)\}$ is obtained by uniformly sampling a band-limited continuous-time signal $x(t)$ with highest frequency $\Omega_{\max}$ at a rate $\Omega_s > 2\Omega_{\max}$, then the original continuous-time signal $x(t)$ can be precisely recovered by passing the impulse train $x_s(t)$ in (1.1) through an ideal (analog) lowpass filter H_r that has a cutoff frequency Ω_c satisfying (1.5). Under these circumstances, an expression that analytically reconstructs the analog signal $x(t)$ with its samples $\{x(nT_s)\}$ can be derived as follows.
From Fig. 1.3, we see that the ideal analog lowpass filter (which is often referred to as the ideal analog *reconstruction filter*) has the frequency response

$$H_r(j\Omega) = \begin{cases} T_s & \text{for } |\Omega| \le \Omega_c \\ 0 & \text{for } |\Omega| > \Omega_c \end{cases}$$

The impulse response of the filter is given by

$$h_r(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H_r(j\Omega)e^{j\Omega t}d\Omega = \frac{T_s}{2\pi}\int_{-\Omega_c}^{\Omega_c} e^{j\Omega t}d\Omega = \frac{\sin(\Omega_c t)}{\Omega_s t/2} \qquad (1.7)$$

The output of the ideal lowpass filter is the convolution of $x_s(t)$ with the impulse response $h_r(t)$. Using (1.2), we compute the filter's output as

$$\hat{x}(t) = \sum_{k=-\infty}^{\infty} x(kT_s) h_r(t - kT_s) \tag{1.8}$$

If we use $\Omega_c = \Omega_s / 2 = \pi / T_s$ and denote $x[k] = x(kT_s)$, then (1.8) becomes

$$\hat{x}(t) = \sum_{k=-\infty}^{\infty} x[k] \cdot \text{sinc}\left[(t - kT_s)/T_s\right] \tag{1.9}$$

where function $\text{sinc}(\bullet)$ is defined by $\text{sinc}(z) = \dfrac{\sin(\pi z)}{\pi z}$. When the sampling rate meets the Nyquist condition (1.4), the continuous-time signal $x(t)$ can be recovered by an ideal lowpass filtering of the sampled signal $x_s(t)$. The formula in (1.9) now shows, in a constructive way, how this recovery can be done using its samples $\{x[k]\}$.

To understand formula (1.9), we view the process of signal reconstruction as superposition of the shifted and scaled impulse response function $x[k]h_r(t - kT_s)$, where

$$h_r(t) = \frac{\sin(\pi t / T_s)}{\pi t / T_s} = \text{sinc}\left(t / T_s\right) \tag{1.10}$$

assumes value 1 at $t = 0$ (by lHôpital's rule) and value 0 at $t = kT_s$ for nonzero integer k. Fig. 1.5 shows a plot of function $h_r(t)$ with $T_s = 1$ over time interval $-10 \le t \le 10$.

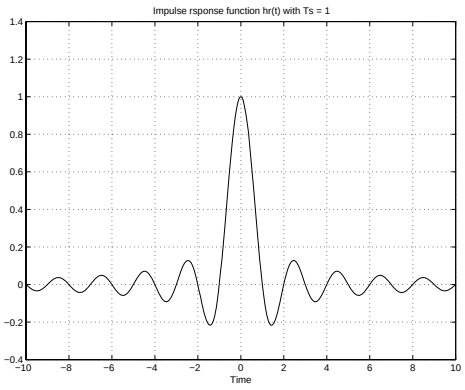


Figure 1.5

C. <u>Sampling of Bandpass Signals</u>

There are applications where the frequencies of the continuous-time signals fall within a higher range $\Omega_L \le \Omega \le \Omega_H$ with $\Omega_L > 0$. Such a signal is referred to as a *bandpass* signal, and the signal's *bandwidth* is defined as $\Delta\Omega = \Omega_H - \Omega_L$. In radio broadcasting, for instance, a (relatively low frequency) audio signal is modulated by a high-frequency carrier and the modulated audio becomes a bandpass signal with a narrow bandwidth. For convenience, below we assume that the highest frequency contained in a bandpass signal $x(t)$ is a multiple of the bandwidth, i.e.,

$$\Omega_H = M \cdot \Delta\Omega \tag{1.11}$$

and choose the sampling frequency Ω_s twice of the signal's bandwidth, i.e.,

$$\Omega_s = 2 \cdot \Delta\Omega = 2\Omega_H / M \tag{1.12}$$

then, according to Eq. (1.3), the Fourier transform of the sampled analog signal $x_s(t)$ becomes

$$X_s(j\Omega) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X(j(\Omega - 2k\Delta\Omega)) \tag{1.13}$$

From Fig. 1.6, we see that the original bandpass signal $x(t)$ can be recovered by appropriately *bandpass filtering* sampled signal $x_s(t)$. Note that, unlike the situation discussed earlier where the frequency content of signal $x(t)$ is in $0 \le \Omega \le \Omega_{\max}$ and it was argued that the sampling frequency must be at least $2\Omega_{\max}$, because M is an integer greater than 1, the sampling frequency set by (1.12) is *less than* $2\Omega_H$. An interpretation of this seemingly contradiction is that the bandwidth in the case of lowpass signals is $\Delta\Omega = \Omega_{\max} - 0 = \Omega_{\max}$, thus the choice $\Omega_s \ge 2\Omega_{\max}$ in the Shannon-Nyquist theorem should be understood as $\Omega_s \ge 2\Delta\Omega$ which is consistent with the one in (1.12).

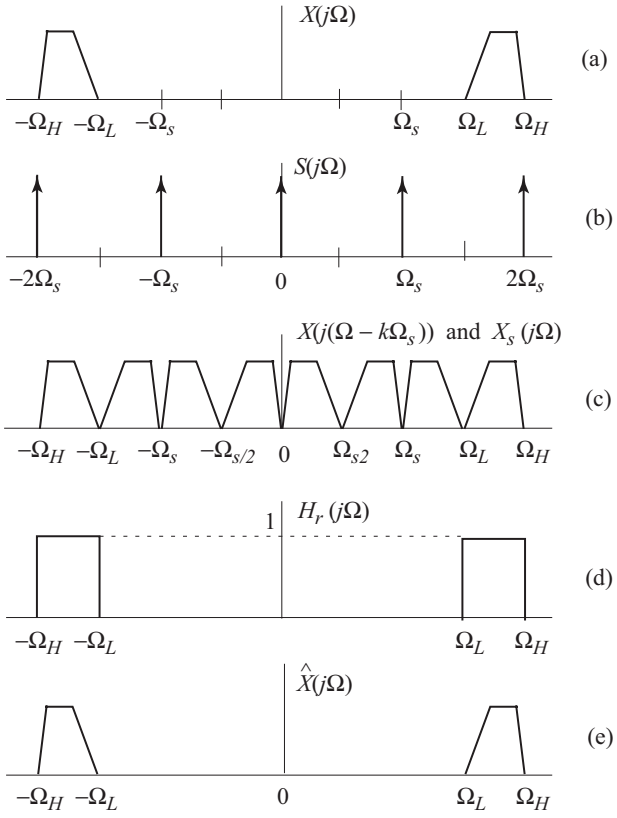


Figure 1.6

1.2 Sparse and Compressible Signals

A discrete signal $\boldsymbol{x}$ of length n is said to be r-sparse, if $\boldsymbol{x}$ contains (at most) r nonzero entries with $r << n$. A signal may not look sparse in a particular domain, but it is very likely that it will look quite sparse in another domain with a proper basis. As an example，we take a look at a two-tone signal that pushing the "A" key on a touch-tone telephone generates:

$$x(t) = \sin(1394\pi t) + \sin(3266\pi t)$$

Fig. 1.7a shows a plot of non-sparse $x(t)$, sampled at 3500 Hz, over the first 36 ms, and Fig. 1.7(b) shows the sparsity of the same signal in the Fourier domain.

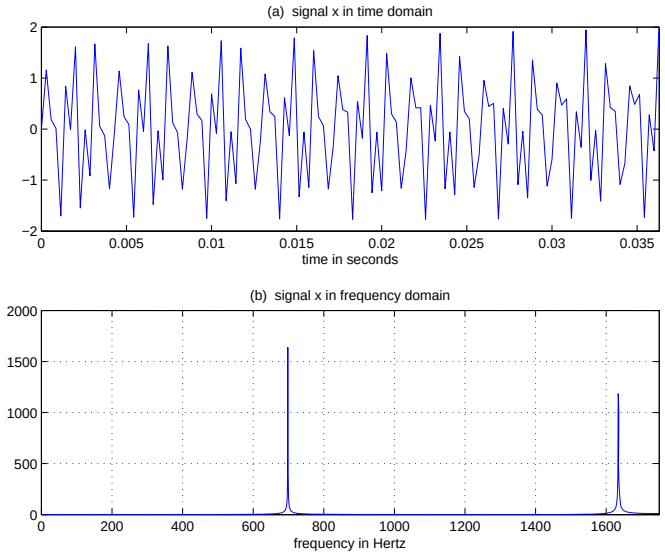


Figure 1.7

An important point to make in this regard is that for natural signals and images, there exist adequate bases and dictionaries (see below) in which signals of interest become sparse or approximately sparse.

A. Compressible Signals

A signal is said to be *approximately sparse* if it differs from a "nearby" sparse signal by a small amount in magnitude. A signal is said to be *compressible*, if there exists a basis in which the signal admits approximately sparse representation. More precisely, signal $\boldsymbol{x}$ is compressible if there exists a basis $\boldsymbol{B}$ in that $\boldsymbol{x}$ becomes nearly sparse: $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{\theta}$ with $\boldsymbol{\theta} = \boldsymbol{\theta}_r + \boldsymbol{w}$ where $\boldsymbol{\theta}_r$ is r-sparse vector that keeps the largest (in magnitude) r entries of $\boldsymbol{\theta}$ and $\boldsymbol{w}$ is small in magnitude. An important point to make here is that meaningful signals always have some kind of structures, and signals with structures are often *compressible*; and many bases exist that are *non-adaptive* in the sense that compressibility property holds for any signals as long as they have structures.

Example 1.1 Recall that the one-dimensional (1-D) discrete cosine transform (DCT) connects a 1-D discrete-time signal of length n, $\{x(i), i = 0, 1, \ldots, n-1\}$, to a length-$N$ sequence $\{c(k), k = 0, 1, \ldots, n-1\}$ via

$$c(k) = \alpha(k)\sum_{i=0}^{n-1} x(i)\cos\left(\frac{(2i+1)k\pi}{2n}\right), \qquad \text{for} \quad 0 \le k \le n-1$$

where

$$\alpha(k) = \begin{cases} \sqrt{1/n} & \text{for } k = 0 \\ \sqrt{2/n} & \text{for } 1 \le k \le n-1 \end{cases}$$

Based on this we can construct an orthonormal basis $\boldsymbol{C}_n$ with its kth row given by

$$\left[\alpha(k-1)\cos[(k-1)\pi/2n] \quad \alpha(k-1)\cos\left[3(k-1)\pi/2n\right] \quad \cdots \quad \alpha(k-1)\cos\left[(2n-1)(k-1)\pi/2n\right]\right]$$

Below is a MALAB code to generate matrix $\boldsymbol{C}_n$.

```
function C = gen_dct(n)
alp = [sqrt(1/n) sqrt(2/n)*ones(1,n-1)];
ind = (1:2:(2*n-1))*pi/(2*n);
C = zeros(n,n);
for k = 1:n,
    C(k,:) = alp(k)*cos((k-1)*ind);
end
```

Consider for instance a gray-scale image of size 64 by 32 taken from the lower-right part of image resolution chart, containing a letter A with three ellipses surrounding it, see Fig. 1.9(a). The matrix associated with that image is denoted by $\boldsymbol{A}$. There are 65 entries in $\boldsymbol{A}$ that are zero. Let $\boldsymbol{a}$ be the vector version of $\boldsymbol{A}$: a = A(:); and $\boldsymbol{C}_{2048}$ be the 1-D DCT matrix of size 2048 by 2048 obtained by C2048 = gen_dct1(2048) , and $\boldsymbol{y}$ be the vector resulted in by applying $\boldsymbol{C}_{2048}$ to $\boldsymbol{a}$: y = C2048*a. Figure 1.8(a) and (b) display signal $\boldsymbol{a}$ and its DCT representation $\boldsymbol{y}$, respectively.

If we set a threshold $\varepsilon_t = 50$, and set the entries in $\boldsymbol{y}$ with magnitude less than ε_t to zeros, then we obtained a vector $\boldsymbol{y}_{1426}$ of size 2048 with a total of 1426 zero entries. Now we can recover the image by applying the inverse DCT to $\boldsymbol{y}_{1426}$ as $\boldsymbol{a}_1 = \boldsymbol{C}_{2048}^T \boldsymbol{y}_{1426}$. The image produced using $\boldsymbol{a}_1$ is shown in Fig. 1.9(b). If one raises the threshold to $\varepsilon_t = 100$, then a $\boldsymbol{y}_{1841}$ can be created in a similarly way with 1841 zero entries. The image produced is shown in Fig. 1.9(c).

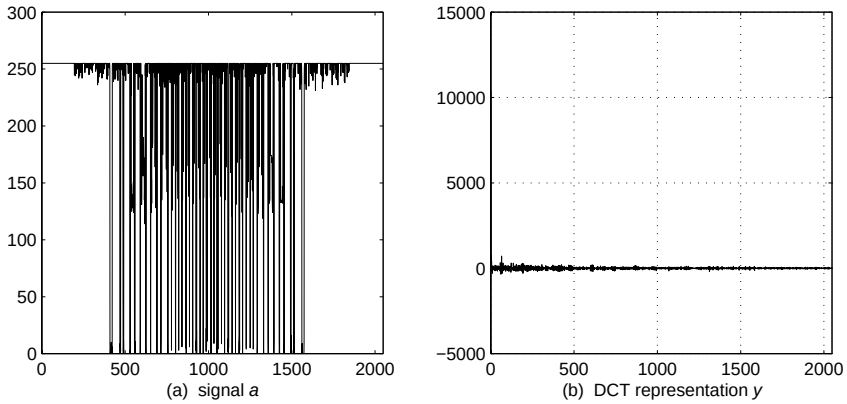
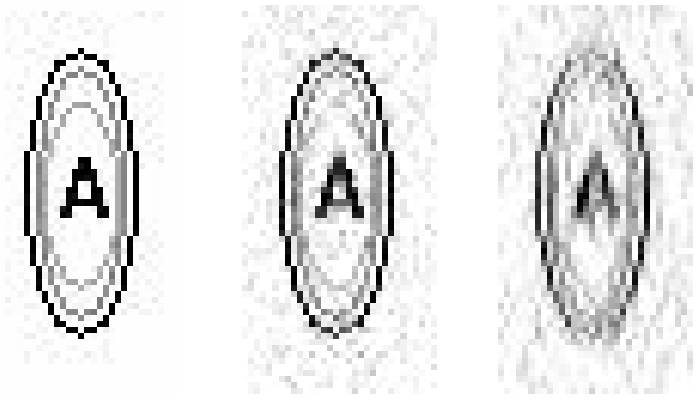


Figure 1.8



(a) original (b) $\varepsilon_t = 50$ (c) $\varepsilon_t = 100$

Figure 1.9

Example 1.2 Our second example is about an orthonormal basis based on discrete wavelet transform (DWT). Suppose signal length $n = 2^p$ for some integer p, then the basis matrix $\boldsymbol{W}$ of size n by n associated with Daubechies' orthogonal discrete wavelets can be readily constructed in MATLAB using Toolbox Uvi_Wave toolbox as follows:

```
% n -- signal length, must be a power of 2.
% L -- length of Daubechies wavelet, must be an even integer.
% W -- orthonormal DWT matrix of size n x n.
function W = gen_wave(n,L)
p = log2(n);
[h0,h1,f0,f1] = daub(L);
I = eye(n,n);
W = I;
for i = 1:n,
    Ii = I(:,i);
    W(:,i) = wt(Ii,h0,h1,p);
end
```

For comparison we consider the same image (and notation) used in Example 1. Let $\boldsymbol{W}_{2048}$ be the wavelet matrix of size 2048 by 2048 associated with the Daubechies D8 wavelet which was obtained by W2048 = gen_wave(2048,8) , and $\boldsymbol{y}$ be the vector obtained by applying $\boldsymbol{W}_{2048}$ to $\boldsymbol{a}$: y = W2048*a. Figures 1.10 (a) and (b) display signal $\boldsymbol{a}$ and its wavelet representation $\boldsymbol{y}$, respectively.

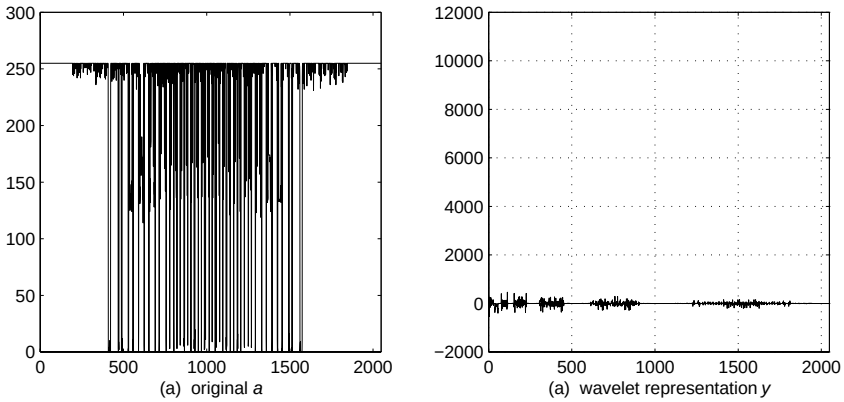
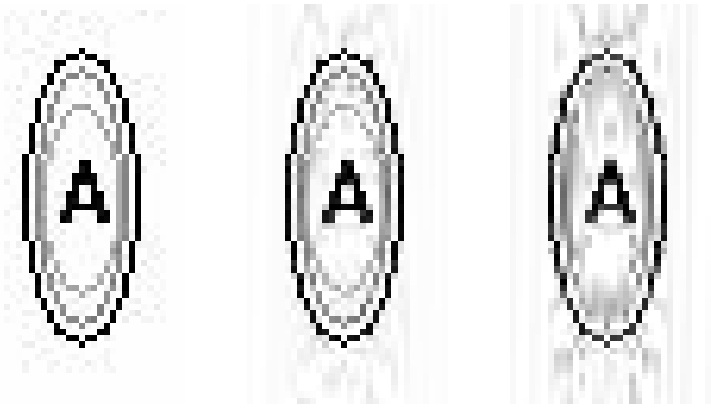


Figure 1.10

Again we set threshold $\varepsilon_t = 50$, and set the entries in $\boldsymbol{y}$ with magnitude less than ε_t to zeros, then we obtained a vector $\boldsymbol{y}_{1631}$ of size 2048 with a total of 1631 zero entries. The image was then recovered by applying the inverse DWT to $\boldsymbol{y}_{1631}$ as $\boldsymbol{a}_1 = \boldsymbol{W}_{2048}^T \boldsymbol{y}_{1631}$. The image produced using $\boldsymbol{a}_1$ is shown in Fig. 1.11(b). With $\varepsilon_t = 100$, a $\boldsymbol{y}_{1847}$ was produced in a similarly way with 1847 zero entries. The image produced is shown in Fig. 1.11(c). The superior reconstruction quality by wavelet D8 relative to that by 1-D DCT is quite evident.



(a) original (b) $\varepsilon_t = 50$ (c) $\varepsilon_t = 100$

Figure 1.11

Example 1.3 For natural images of relatively large size, a matrix representation of DWT will be of extremely large in size, making it impractical to use or store. However, DWT can be implemented in a multiresolution system using for example free-type filter banks without expressing it as a matrix, hence sparse representation of large image can be done with ease [1].
As an example, we applied 2-D DWT with D8 to image boat512 of size 512 by 512. The DWT result is a matrix $\boldsymbol{Y}$ of same size and is shown in Fig. 1.12b, which is very sparse (a bright pixel represents a small numerical value and a dark pixel represents a large value).

(a) (b)

Figure 1.12

Then an $\varepsilon_t = 35$ was used to hard-threshold $\boldsymbol{Y}$ that yields a matrix $\boldsymbol{Y}_1$ with 250195 zero valued entries. Note that $(512^2 - 250195)/512^2 = 0.0456$ (note: $512^2 = 262144$), that is, more than 95% of the entries in $\boldsymbol{Y}_1$ are zero. The inverse DWT with D8 was applied to $\boldsymbol{Y}_1$ to yield an image which is

shown in Fig. 1.13b.

Figure 1.13

```
[h0,h1,f0,f1] = daub(8);
Y = wt2d(boat512,h0,h1,9);
ind = find(abs(Y) < 35);
Y1 = Y;
Y1(ind) = 0;
X1 = iwt2d(Y1,f0,f1,9);
map = gray(256);
subplot(121)
imshow(boat512,map)
subplot(122)
imshow(X1,map)
```

B. *Overcomplete Dictionaries*

Like a basis in a finite-dimensional vector space can be described by a square matrix $\boldsymbol{B}$, an (*over-complete*) *dictionary* can be characterized by a matrix $\boldsymbol{D} \in C^{n \times l}$ with $l > n$ where the l (column) vectors contain at least n linearly independent vectors (hence if one wishes these vectors form a basis). Obviously a dictionary is *bigger* than a basis as it also contains more than necessary many vectors to represent signals in the space, hence the term "overcomplete". Each (column) vector in a dictionary is called an *atom*. An important observation on dictionaries is that given an (overcomplete) dictionary $\boldsymbol{D} \in C^{n \times l}$ with $l > n$ and a signal $\boldsymbol{x}$ of length n, representations of signal $\boldsymbol{x}$ in $\boldsymbol{D}$ in terms of linear combinations of some atoms are *not* unique. This is to say that if $\boldsymbol{D}$ contains more than n nonzero atoms and if $\boldsymbol{D}$ is of full row rank, then the underdetermined system of linear equations $\boldsymbol{D\theta} = \boldsymbol{x}$ admits infinitely many solutions. Among these representations of $\boldsymbol{x}$, we are particularly interested in finding the most economical one, $\boldsymbol{\theta}^*$, that is the sparsest.

Examples of Dictionaries

- $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{C}_n^T \end{bmatrix}$, where $\boldsymbol{I}_n$ is the identity matrix of size $n \times n$, $\boldsymbol{C}_n$ is the 1-D DCT matrix of size $n \times n$.
- $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{W}_n^T \end{bmatrix}$, where $\boldsymbol{W}_n$ is a 1-D DWT matrix.

• $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{F}_n^H \end{bmatrix}$, where $\boldsymbol{F}_n$ is the matrix of size $n \times n$ constructed by 1-D discrete Fourier transform (DFT). Recall that the DFT connects a 1-D discrete-time signal of length n, $\{x[i], i = 0, 1, \ldots, n-1\}$, to a length-$n$ sequence $\{X(k), k = 0, 1, \ldots, n-1\}$ via

$$X[k] = \frac{1}{\sqrt{N}} \sum_{i=0}^{n-1} x(i) W_n^{ki}, \quad W_n = e^{-j2\pi/n} \quad \text{for} \quad 0 \le k \le n-1$$

If we define $\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T$ and $\boldsymbol{X} = \begin{bmatrix} X(0) & X(1) & \cdots & X(N-1) \end{bmatrix}^T$, then $\boldsymbol{X} = \boldsymbol{F}_n \boldsymbol{x}$ where

$$\boldsymbol{F}_n = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_n^1 & W_n^2 & \cdots & W_n^{n-1} \\ 1 & W_n^2 & W_n^4 & \cdots & W_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_n^{n-1} & W_n^{2(n-1)} & \cdots & W_n^{(n-1)\times(n-1)} \end{bmatrix}$$

Note that matrix $\boldsymbol{F}_n$ is unitary, i.e. $\boldsymbol{F}_n \cdot \boldsymbol{F}_n^H = \boldsymbol{F}_n^H \cdot \boldsymbol{F}_n = \boldsymbol{I}_n$.

• $\boldsymbol{D} = [\boldsymbol{I}_n \ \ \hat{\boldsymbol{H}}_n]$ where $\hat{\boldsymbol{H}}_n = \boldsymbol{H}_n / n^{1/2}$ with $\boldsymbol{H}_n$ being the Hadamard-Walsh matrix of $n \times n$ which can be generated recursively as

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad \boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{H}_{n/2} & \boldsymbol{H}_{n/2} \\ \boldsymbol{H}_{n/2} & -\boldsymbol{H}_{n/2} \end{bmatrix}$$

and $n = 2^p$ for some integer $p > 0$. Note that $\hat{\boldsymbol{H}}_n$ is an orthogonal matrix.

```
function H = gen_hw(p)
n = 2^p;
H = 1;
for i = 1:p,
    H = [H H; H -H];
end
H = H/sqrt(n);
```

Definition: Given a vector $\boldsymbol{x} = [x_1 \ x_2 \cdots x_n]^T$, the l_p norm of $\boldsymbol{x}$ for $p \ge 1$ is defined by

$$\|\boldsymbol{x}\|_p = \left[\sum_{i=1}^{n} |x_i|^p \right]^{1/p}$$

In particular, $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$, $\|\boldsymbol{x}\|_2 = \left[\sum_{i=1}^{n} |x_i|^2 \right]^{1/2}$, $\|\boldsymbol{x}\|_\infty = \max_{1 \le i \le n} \{|x_i|\}$.

Recall that for $\|\bullet\|$ to be a vector norm, four conditions must be satisfied:

(i) $\|\boldsymbol{x}\| \ge 0$

(ii) $\|\boldsymbol{x}\| = 0 \quad \Leftrightarrow \quad \boldsymbol{x} = \boldsymbol{0}$

(iii) $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\|$

(iv) $\|\boldsymbol{x} + \boldsymbol{y}\| \le \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$

It can be readily verified that the l_p norm satisfies these conditions as long as $p \ge 1$, but condition (iv) fails for $0 < p < 1$. If we take the l_p norm as a function of $\boldsymbol{x}$, i.e. $f(\boldsymbol{x}) = \left[\sum_{i=1}^{n} |x_i|^p \right]^{1/p}$, then $f(\boldsymbol{x})$ is convex as long as $p \ge 1$, and $f(\boldsymbol{x})$ is nonconvex for $p < 1$. In the literature, the sparsity of a discrete signal (vector) $\boldsymbol{x}$ is often related to the so-called l_0 norm defined by

$$\|\boldsymbol{x}\|_0 = \sum_{i=1}^{n} |x_i|^0 = \text{number of nonzero components in } \boldsymbol{x}$$

We stress that this "l_0 norm" is *not* a norm as it violates conditions (iii) and (iv) -- it merely is the number of nonzero entries in $\boldsymbol{x}$. However the notion is quite convenient and intuitive, indeed the problem of finding the sparsest solution of the underdetermined linear system $\boldsymbol{D\theta} = \boldsymbol{x}$ can now be cast as

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{\theta}\|_0 \\ &\text{subject to:} && \boldsymbol{D\theta} = \boldsymbol{x} \end{aligned} \tag{1.14}$$

It turns out that (1.14) is a problem of combinatorial complexity: finding solution of (1.14) requires enumerating subsets of the dictionary to identify the smallest subset that can represent signal $\boldsymbol{x}$, the complexity of such a subset search grows *exponentially* with l. An important result concerning sparse signals and compressed sensing is that under certain conditions the sparsest solution of $\boldsymbol{D\theta} = \boldsymbol{x}$ can be obtained by solving the convex problem

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{\theta}\|_1 \\ &\text{subject to:} && \boldsymbol{D\theta} = \boldsymbol{x} \end{aligned} \tag{1.15}$$

If the data involved are all real-valued, (1.15) is equivalent to a linear program (LP), if the data are complex-valued, then (1.15) is essentially a second-order cone program (SOCP). Hence (1.15) can be solved effectively using any good convex program solvers like SeDuMi.

Example 1.4 In this example, we consider the problem of representing a discrete signal $\boldsymbol{x}$ by sparse linear combination of atoms of a dictionary. The signal under consideration is of length 256 and is generated by sampling continuous-time signal $e^{-5t}\sin(80.653t)$ at a rate of 255 Hz over time interval [0, 1] and then adding a total of 11 spikes of various magnitudes at various sampling instants to it. The signal is depicted in Fig. 1.14 and the MATLAB code used is given below.

```
% Example x = gen_ex1_4(256,11,1,45);
function x = gen_ex1_4(n,r,st1,st2)
randn('state',st1)
rand('state',st2)
x1 = zeros(n,1);
q = randperm(n);                % random permutation
x1(q(1:r)) = 0.5*randn(r,1);
```

```
t = 0:1/(n-1):1;
x2 = exp(-5*t).*sin(80.653*t);
x = x1(:) + x2(:);
plot(t,x)
```

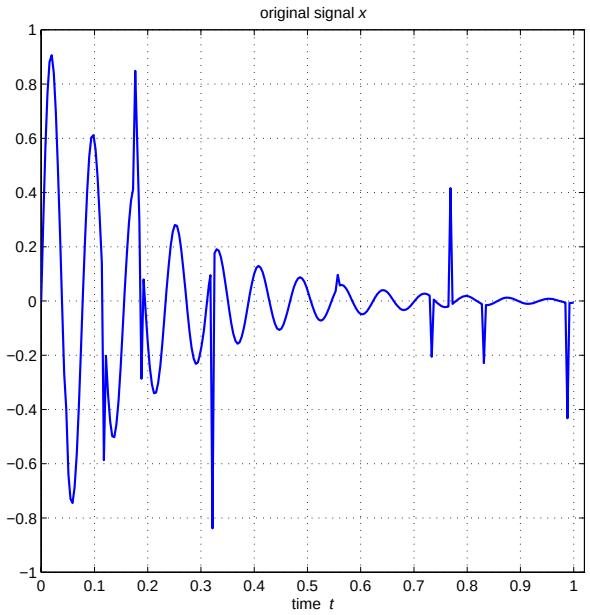


Figure 1.14

We use $\boldsymbol{D} = \left[\boldsymbol{I}_n \quad \boldsymbol{C}_n^T\right]$ with n = 256 as our dictionary and solve problem (1.15) as an LP, see the MATLAB code (which requires toolbox SeDuMi) below .

```
function a = sol_ex3(D,x)
[n,m] = size(D);
I = eye(n);
Z = zeros(n,n);
Ct = psi(:,(n+1):m);
A = [-I -I Z; I -I Z; Ct Z -I; -Ct Z -I]';
c = [zeros(2*n,1); x; -x];
b = [zeros(n,1); -ones(m,1)];
[xs,ys,info] = sedumi(A,b,c);
info
a2 = ys(1:n);
a1 = x - Ct*a2;
a = [a1; a2];
```

The solution is a vector $\boldsymbol{\theta}$ of length 512 which is displayed in Fig. 1.15. From the figure we see that the solution is quite sparse. By hard-thresholding $\boldsymbol{\theta}$ with an $\varepsilon_t = 0.1$, we obtained an even sparser $\hat{\boldsymbol{\theta}}$ with 468 zero entries. If we use $\hat{\boldsymbol{x}} = \boldsymbol{D}\hat{\boldsymbol{\theta}}$ to reconstruct signal $\boldsymbol{x}$, the error is found to be $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 / \|\boldsymbol{x}\|_2 = 0.0558$. Signal $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$ are shown in Fig. 1.16.

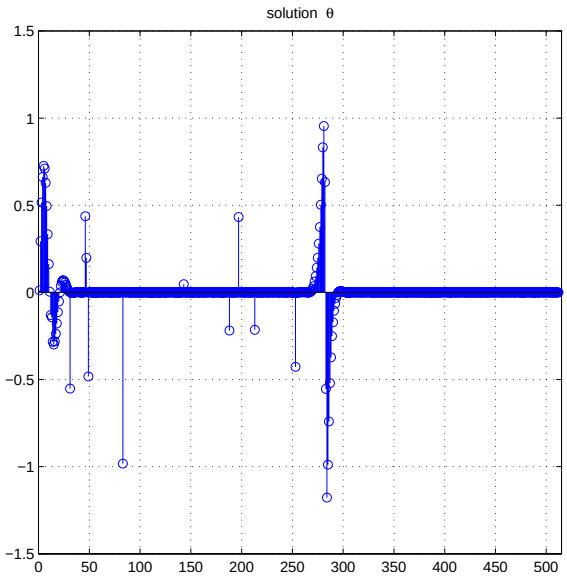


Figure 1.15

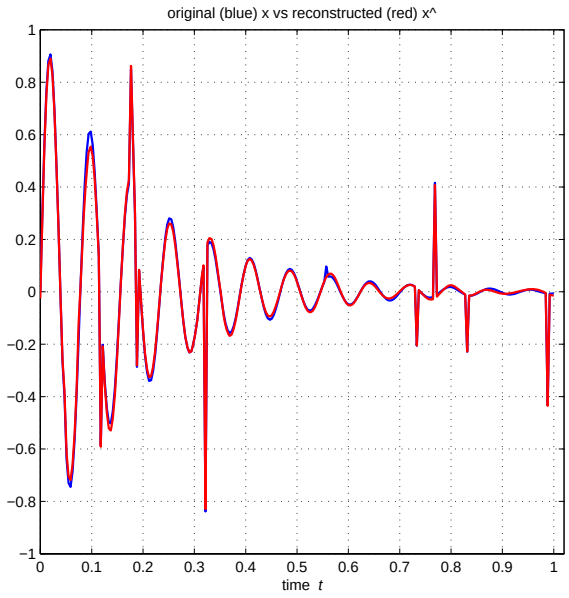


Figure 1.16

We note that among 512 entries of $\hat{\boldsymbol{\theta}}$, there are only 44 nonzero components. If we give up using dictionary $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{C}_n^T \end{bmatrix}$, and use an orthogonal basis, say wavelet basis $\boldsymbol{W}_8$ built on Daubechies wavelet D8 (with n = 256), instead, then the signal $\boldsymbol{x}$ is represented by $\boldsymbol{\theta}_8 = \boldsymbol{W}_8 \boldsymbol{x}$. Fig. 1.17 shows vector $\boldsymbol{\theta}_8$ which displays certain sparseness but not as sparse as vector $\boldsymbol{\theta}$ shown in Fig. 1.15.

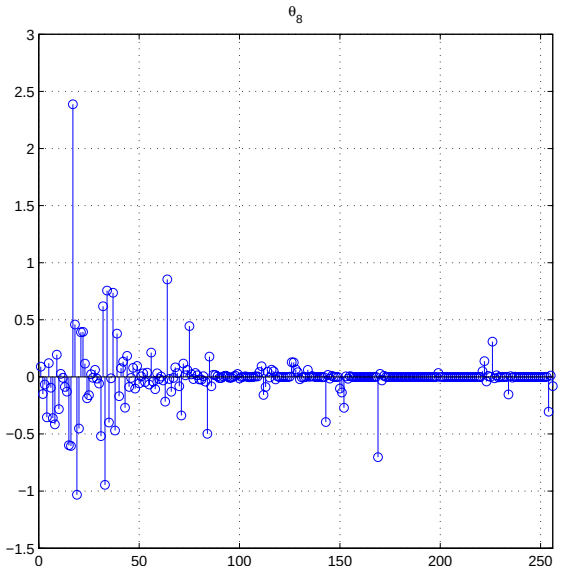


Figure 1.17

We used $\varepsilon_t = 0.137$ to hard-threshold $\boldsymbol{\theta}_8$, 212 zeros were generated with 44 nonzero entries left there. We denote such sparse approximation of $\boldsymbol{\theta}_8$ by $\hat{\boldsymbol{\theta}}_8$, and use it to reconstruct the signal as $\hat{\boldsymbol{x}}_8 = \boldsymbol{W}_8^T \hat{\boldsymbol{\theta}}_8$. The reconstruction error was found to be $\|\boldsymbol{x} - \hat{\boldsymbol{x}}_8\|_2 / \|\boldsymbol{x}\|_2 = 0.1522$ (compared to 0.0558 achieved by dictionary $\boldsymbol{D} = \left[\boldsymbol{I}_n \quad \boldsymbol{C}_n^T\right]$). Fig. 1.18 shows the reconstructed signal $\hat{\boldsymbol{x}}_8$ versus the original $\boldsymbol{x}$.

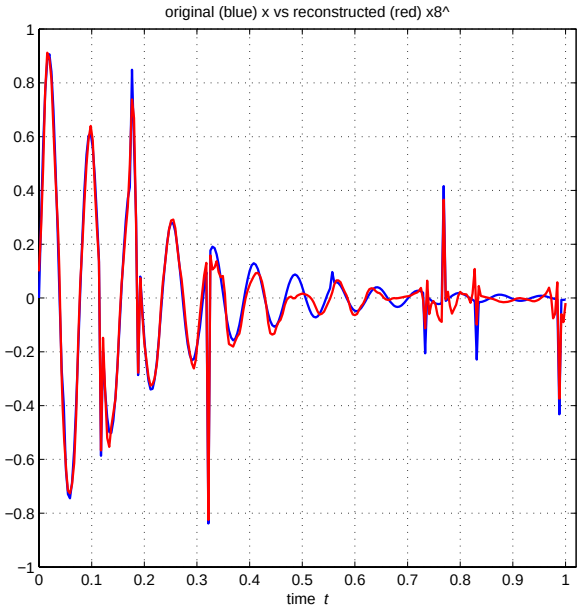


Figure 1.18

C. ***Coherence Between Two Bases***

Typically a dictionary $\boldsymbol{D}$ is composed of several bases: $\boldsymbol{D} = \left[\boldsymbol{\Psi}_1 \quad \boldsymbol{\Psi}_2 \quad \cdots \quad \boldsymbol{\Psi}_q\right]$ with each $\boldsymbol{\Psi}_i$ is a

basis. The case of $q = 2$ is of particular interest, not only because it is the simplest case of a nontrivial dictionary, but also because coherence between *two* bases is closely related to the CS theory, as will be seen in Part II. For notation simplicity, here we consider a dictionary $\boldsymbol{D} = [\boldsymbol{\Phi} \quad \boldsymbol{\Psi}]$ where $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are two orthonormal basis matrices of size n by n. The *coherence*, which is also known as *mutual coherence* in the literature, between $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ is defined as

$$\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \sqrt{n} \cdot \max_{1 \le k, j \le n} \left| \boldsymbol{\varphi}_k^T \boldsymbol{\psi}_j \right| \tag{1.16}$$

where $\boldsymbol{\varphi}_k^T$ and $\boldsymbol{\psi}_j$ are the kth row and jth column of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, respectively. Obviously μ measure the *largest correlation* between an atom in basis $\boldsymbol{\Phi}$ and an atom in basis $\boldsymbol{\Psi}$. Using Cauchy-Schwarz inequality, we get $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \le \sqrt{n}$. On the other hand, $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \ge 1$ must hold, because otherwise we would have $\left| \boldsymbol{\varphi}_k^T \boldsymbol{\psi}_j \right| < 1/\sqrt{n}$ implying that $\|\boldsymbol{\varphi}_k\|^2 = \sum_{j=1}^{n} \left| \boldsymbol{\varphi}_k^T \boldsymbol{\psi}_j \right|^2 < 1$ which contradicts with $\boldsymbol{\varphi}_k$ being a unit vector. Therefore the coherence measure μ in (1.16) is always in the range $\left[1, \sqrt{n}\right]$.

We are interested in bases pairs with *small* coherence and in that case we say the two bases are *incoherent*. In the case of dictionaries for sparse representation, the incoherence is important because a small μ means small correlation between the two bases, thus a dictionary composed by such a pair of bases has a "richer vocabulary" relative to a dictionary with a larger μ. In the context of CS, the incoherence between the basis involved in signal sensing and the basis involved in signal representation/reconstruction is a crucial feature of an effective CS system (see part II). An important question to address is what kind of matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ should be in order for them to have a low mutual coherence? If we let $\boldsymbol{U} = \sqrt{n} \cdot \boldsymbol{\Phi}\boldsymbol{\Psi} = \{u_{k,j}\}$, then the kth row of $\boldsymbol{U}$ is equal to $\sqrt{n} \cdot \boldsymbol{\varphi}_k^T \boldsymbol{\Psi}$ whose 2-norm is $\sqrt{n}$. Therefore, to possess a small μ, the entries in each row of $\boldsymbol{U}$ must not be concentrated but wide spread. For instance, a most concentrated row of $\boldsymbol{U}$ would look like $\left[0 \quad \cdots \quad 0 \quad \pm\sqrt{n} \quad 0 \quad \cdots \quad 0\right]^T$ which gives $\mu = \sqrt{n}$, while a most "spread out" row of $\boldsymbol{U}$ would look like $\left[\pm 1 \quad \pm 1 \quad \cdots \quad \pm 1\right]^T$ which would lead to $\mu = 1$ if every row of $\boldsymbol{U}$ is like that. Similar claim can be made for the columns of $\boldsymbol{U}$. Furthermore, requiring $\sqrt{n} \cdot \boldsymbol{\varphi}_k^T \boldsymbol{\Psi} = \left[\pm 1 \quad \pm 1 \quad \cdots \quad \pm 1\right]^T$ for each row of $\boldsymbol{\Phi}$ is the same as saying that each row of $\boldsymbol{\Phi}$ must be "spread out" in the $\boldsymbol{\Psi}$ domain. The same can also be said about $\boldsymbol{\Psi}$: each column of $\boldsymbol{\Psi}$ must be spread out in the $\boldsymbol{\Phi}$ domain in order to have a small mutual coherence.

Examples

- $\mu\left(\boldsymbol{I}_n, \boldsymbol{C}_n^T\right) = \sqrt{2}$
- $\mu\left(\boldsymbol{I}_n, \boldsymbol{F}_n^H\right) = 1$
- $\mu\left(\boldsymbol{I}_n, \hat{\boldsymbol{H}}_n\right) = 1$ where $\hat{\boldsymbol{H}}_n = \boldsymbol{H}_n / n^{1/2}$ with $\boldsymbol{H}_n$ being the Hadamard-Walsh matrix
- $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$ where $\boldsymbol{\Psi}$ is *any* fixed orthonormal basis and $\boldsymbol{\Phi}$ is generated by orthonormalizing n vectors sampled independently and uniformly on the unit sphere. Then with high probability $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \approx \sqrt{2 \log n}$.
- $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$ is also very low where $\boldsymbol{\Psi}$ is any fixed orthonormal basis and $\boldsymbol{\Phi}$ is an orthonormal matrix with independent identically distributed (i.i.d.) entries, e.g. Gaussian or ± 1 binary entries.

II. Compressed Sensing and Signal Reconstruction

2.1 Sensing a Sparse Signal

Although the concept of signal sparsity and l_1-norm based recovery techniques can be traced back to the work of Logan in 1965, Santosa and Symes in 1986, and Donoho and Stark in 1989, it is generally agreed that the foundation of today's CS theory was laid by the three papers [2]-[4] in 2006 that have inspired a burst of intensive research activities in CS in the past five years.

In a nutshell, compressive sensing (CS) acquires a signal of interest *indirectly* by correcting a *small* number of its "projections" rather than evenly sampling it at the Nyquist rate which can be prohibitively high for broadband signals encountered in many applications. This new signal acquisition paradigm has revolutionized the way digital data are traditionally acquired. Before we proceed with technical details, it should be stressed that CS is an effective solution approach in situations such as these [5]:

- The number of sensors are limited due to implementation constraints or cost;
- Measurements may be extremely expensive as in certain image processing via neutron scattering;
- The sensing process may be slow so that only a small number of measurements can be collected.
- Many inverse problems are such that the only way to acquire data is to use a measurement matrix. Hence the models used in inverse problems make them an excellent application domain for CS.
- The conventional Nyquist rate of sampling is too high to implement. For example the current analog-to-digital conversion (ADC) technology based on uniform sampling in the time or spatial domain is limited to within the order of 1 GHz.

The key notions in the development of current CS theory are *sparsity* and *incoherence*. Both concepts were introduced and briefly discussed in part I. Reference [5] highlights the principle of sparsity by saying that

- Sparsity expresses the idea that the "*information rate*" of a continuous-time signal may be much smaller than that suggested by its bandwidth;
- A discrete-time signal depends on a number of degrees of freedom which is relatively much smaller than its (finite) length; and
- Many natural signals are sparse or compressible in the sense that they have sparse or

approximately sparse representations when expressed in an appropriate basis or dictionary Ψ.

References [5], [6] explain the notion of incoherence between a sensing matrix Φ and a sparsifying basis Ψ by saying that

- The test vectors (i.e. the rows in Φ) must be spread out in the Ψ domain, just as a spike in the time domain is spread out in the frequency domain;
- In this regard, incoherence extends the duality between time and frequency.

Now let us consider a discrete signal $\boldsymbol{x}$ which itself may or may not be sparse in the canonical basis but is sparse or approximately sparse in an appropriate basis Ψ. That is,

$$\boldsymbol{x} = \Psi\boldsymbol{\theta} \tag{2.1}$$

where $\boldsymbol{\theta}$ is sparse or approximately sparse. A central idea in the current CS theory is about how a (discrete) signal is acquired: the acquisition of signal $\boldsymbol{x}$ of length n is carried out by measuring m projections of $\boldsymbol{x}$ onto sensing (also known as testing) vectors $\{\boldsymbol{\varphi}_i^T, i = 1,2,...,m\}$: $y_i = \boldsymbol{\varphi}_i^T\boldsymbol{x}$ for i = 1, 2, …, m. For sensing efficiency, one wishes to collect a relatively *much smaller* number of measurements, that is, one requires that m be considerably smaller than n, hence the name *compressed sensing*. This data acquisition mechanism is at the core of a CS system that marks a fundamental departure from the conventional data acquisition-compression-transmission-decompression framework: the conventional framework collects a vast amount of data for acquiring a high-resolution signal or image, then essentially discard most of the data collected (in the Ψ domain) in the compression stage, while in CS the data are measured in an compressed manner, and the much reduced amount of measurements are transmitted or stored economically, and every bit of the measurements are then utilized to recover the signal at the base station where sufficient computing resources are available to carry out the reconstruction algorithm. Using matrix notation the sensing process is described by

$$\boldsymbol{y} = \hat{\Phi}\cdot\boldsymbol{x} \tag{2.2}$$

where

$$\hat{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_1^T \\ \boldsymbol{\varphi}_2^T \\ \vdots \\ \boldsymbol{\varphi}_m^T \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

and for meaningful projections, it is often assumed that the lengths of the projection vectors $\{\boldsymbol{\varphi}_i^T, i = 1,2,...,m\}$ are unity. Here are two questions that naturally arise from signal model (2.2)

(i) What type of matrix $\hat{\Phi}$ should one choose for the purpose of sensing? and

(ii) How many measurements $\{y_i, i = 1, 2, \ldots, m\}$ should one collect so that these measurements will be sufficient to recover signal $\boldsymbol{x}$?

The following theorem addresses these questions explicitly.

Theorem 1 [6]

Given $\boldsymbol{x} \in R^n$ and suppose $\boldsymbol{x}$ is r-sparse in basis $\boldsymbol{\Psi}$. Select m measurements in the $\boldsymbol{\Phi}$ domain uniformly at random via (2.2) (that is, the m testing vectors $\{\boldsymbol{\varphi}_i^T, i = 1,2,...,m\}$ are m rows uniformly randomly selected from matrix $\boldsymbol{\Phi}$). If

$$m \geq C \cdot \mu^2(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \cdot r \cdot \log n \tag{2.3}$$

for some positive constant C, then signal $\boldsymbol{x}$ can be exactly reconstructed with overwhelming probability by (2.1), where $\boldsymbol{\theta}$ solves the convex l_1 minimization problem

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{\theta}\|_1 \\ &\text{subject to:} \quad \hat{\boldsymbol{\Phi}}\boldsymbol{\Psi}\boldsymbol{\theta} = \boldsymbol{y} \end{aligned} \tag{2.4}$$

Remarks

(i) It is important to understand the significance of condition (2.3). This condition relates all the key parameters in a CS platform: the signal size n, its sparsity K (in domain $\boldsymbol{\Psi}$), the number of measurements m that is sufficient to recover x, and the incoherence between $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, and it says that one can use a number of measurements m that is substantially smaller the signal's dimension n to reconstruct the signal if bases $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are incoherent, i.e. $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \approx 1$. Indeed if $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = 1$, (2.3) becomes

$$m \geq C \cdot r \cdot \log n \tag{2.5}$$

Even better, there is a *four-to-one* practical rule which says that for exact reconstruction, one needs about *four incoherent measurements per unknown nonzero term* in $\boldsymbol{x}$:

$$m \geq 4r \tag{2.6}$$

regardless of the signal's dimension n [5]. We stress that to reach condition (2.5) or the 4-to-1 rule in (2.6), the low coherence is the key. In effect, if this does not hold, say $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \sqrt{n}$, then (2.3) becomes

$$m \geq C \cdot n \cdot r \cdot \log n$$

Obviously, in this case no compressed sensing will be achieved.

(ii) Let $\boldsymbol{A} = \hat{\boldsymbol{\Phi}}\boldsymbol{\Psi}$ so we can write problem (2.4) as

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{\theta}\|_1 \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{y} \end{aligned} \tag{2.7}$$

The constraint $\boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{y}$ in (2.7) is an *underdetermined* linear system who has more unknowns than equations. Therefore, finding a sparse solution of $\boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{y}$ is an *ill-posed* problem. Fig. 2.1 offers an

intuitive explanation of why minimizing l_1 norm of the solutions of $\boldsymbol{A\theta} = \boldsymbol{y}$ helps obtain a sparse solution.

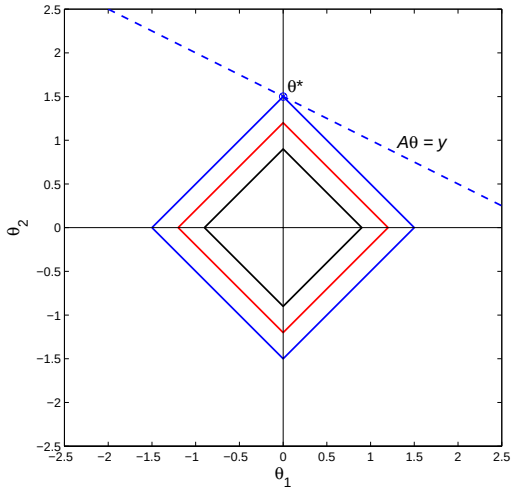


Figure 2.1

On the other hand, Fig. 2.2 explains why minimizing l_2 norm fails to offer a sparse solution.

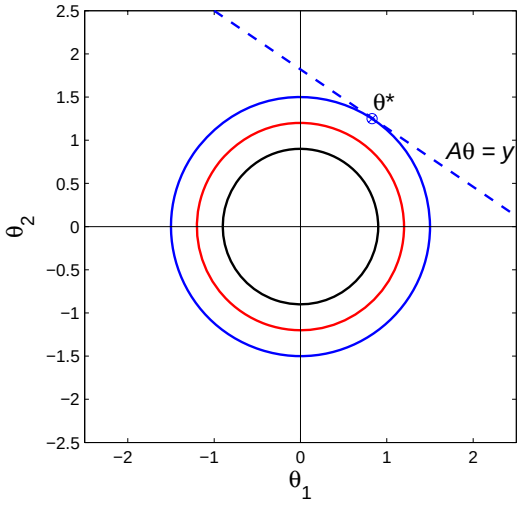


Fig. 2.2

Having determined $\boldsymbol{\Psi}$ and $\hat{\boldsymbol{\Phi}}$, and collected measurements $\boldsymbol{y}$ via (2.2), one needs to solve l_1 minimization problem (2.4). Suppose the data involved in (2.4) are all real valued. Denote $\boldsymbol{A} = \hat{\boldsymbol{\Phi}}\boldsymbol{\Psi}$ and let the upper bound for the ith component of $\boldsymbol{\theta}$ be d_i. The (2.4) is converted to a linear programming (LP) problem as follows:

$$\begin{aligned} \underset{\boldsymbol{\theta},\boldsymbol{d}}{\text{minimize}} \quad & \sum_{i=1}^{n} d_i \\ \text{subject to:} \quad & -\boldsymbol{d} \le \boldsymbol{\theta} \le \boldsymbol{d} \\ & \boldsymbol{A\theta} = \boldsymbol{y} \end{aligned} \tag{2.8}$$

where both $\boldsymbol{\theta}$ and $\boldsymbol{d} = [d_1\ d_2\ \ldots\ d_n]^T$ are treated as unknowns. Once the solution $\boldsymbol{\theta}^*$ of (2.8) is

found, the signal x is reconstructed as $\boldsymbol{x}^* = \boldsymbol{\Psi}\boldsymbol{\theta}^*$.

Example 2.1 We demonstrate the signal sampling and recovery techniques with a discrete-time signal $\boldsymbol{x}$ of length $n = 1024$ with sparsity $r = 50$ which was generated as follows.

```
n = 1024;
r = 50;
x = zeros(n,1);
rand('state',4)
q = randperm(n);
randn('state',17)
x(q(1:r)) = randn(r,1);
```

Signal $\boldsymbol{x}$ is shown in Fig. 2.3 in blue dots. A random sensing matrix $\hat{\boldsymbol{\Phi}}$ with $m = 200$ was generated as

```
m = 200;
Phi = randn(m,n);
Phi = orth(Phi')';
Phi = (sqrt(n))*eye(m)*Phi;
```

and the l_1 minimization was performance using l_1-MAGIC by a MATLAB toolbox for CS produced by CalTech:

```
y = Phi*x;
A = Phi;
x0 = A'*y;
xh = l1eq_pd(x0,A,[],y,1e-3);
```

The result is a reconstructed signal $\hat{\boldsymbol{x}}$ (xh in the code) which is shown in Fig. 2.3 as red circles. We observe that $\hat{\boldsymbol{x}}$ recovers the original $\boldsymbol{x}$ perfectly.

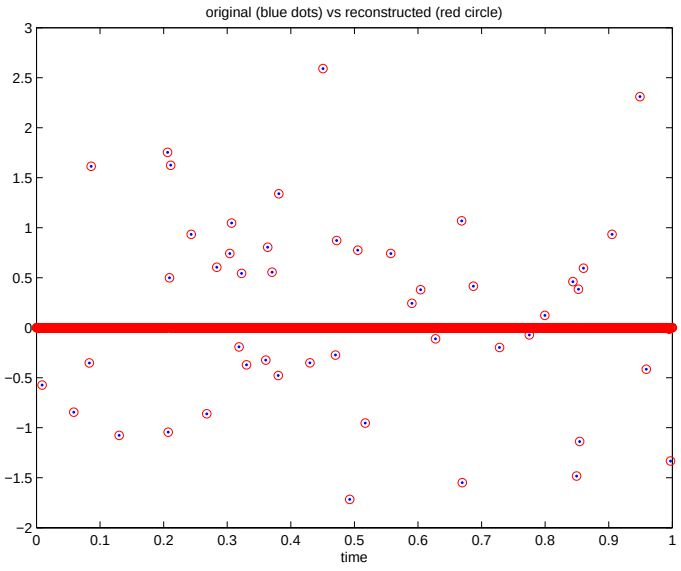


Figure 2.3

We note that in the above the ratio of m and r is consistent with the 4-to-1 rule. If less number of samples are collected, the recovery may be inaccurate can be seen in Figure 2.4 which shows a reconstruction outcome with $r = 50$ and $m = 193$.

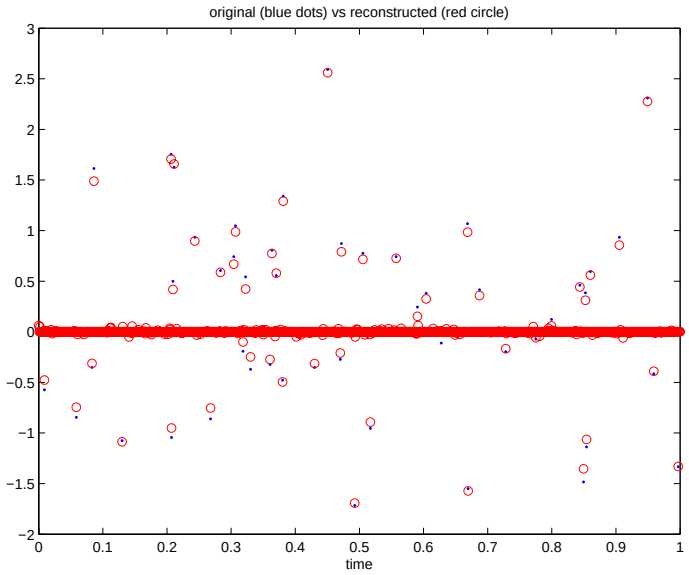


Figure 2.4

Example 2.2 Next we consider a signal which is not sparse in the time domain but sparse in the frequency domain. The signal was created as follows. Let

$$x(t) = \sin(697\pi t) + \sin(1975\pi t)$$

Signal $x(t)$ was sampled at 2000 Hz and the first 1024 samples was used for generate a discrete-time signal $\boldsymbol{x}_0$:

```
n = 1024;
dt = 1/2000;
t = 0:dt:((n-1)*dt);
t = t(:);
x0 = sin(697*pi*t) + sin(1975*pi*t);
```

Next, a1024-point DCT was applied to $\boldsymbol{x}_0$ to obtain $\boldsymbol{c}_0$ where every entry with magnitude less than 0.25 was set to zero. The DCT coefficients so modified contains 939 zeros, then the sparsity $r = 85$. The 1024-point inverse DCT was then applied to the modified DCT coefficients and obtain a discrete-time signal $\boldsymbol{x}$. By the construction, $\boldsymbol{x}$ in sparse in the DCT domain with $r = 85$, but not sparse in the time domain. In effect, the difference between signals $\boldsymbol{x}_0$ and $\boldsymbol{x}$ are insignificant as can be observed from Fig. 2.5, where 51 samples of $\boldsymbol{x}$ are compared against those of $\boldsymbol{x}_0$.

```
Dn = gen_dct(n);
y0 = Dn*x0;
ind = find(abs(y0) < 0.25);
r = n - length(ind); % sparsity r = 85
y1 = y0;
y1(ind) = 0;
x = Dn'*y1;
```

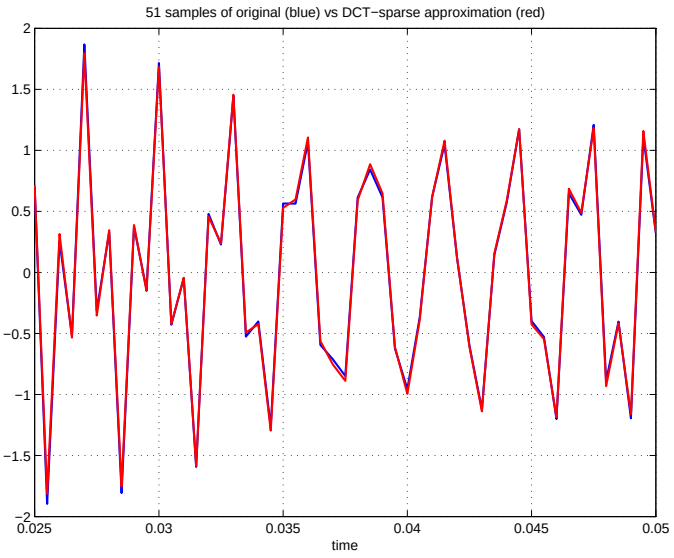


Figure 2.5

An m by n matrix $\hat{\boldsymbol{\Phi}}$ with m = 300 was constructed by randomly selecting m rows from the identity matrix $\boldsymbol{I}_n$. This means that the measurement $\boldsymbol{y}$ was obtained by randomly sampling signal $\boldsymbol{x}$. The inverse DCT matrix $\boldsymbol{D}_n^T$ was used as basis $\boldsymbol{\Psi}$, note that in this case $\mu(\boldsymbol{\Phi},\boldsymbol{\Psi}) = \sqrt{2}$.

```
rand('state',15)
q = randperm(n);
q = q(:);
y = x(q(1:m));
Psi1 = Dn';
A = Psi1(q(1:m),:);
theta_0 = A'*y;
theta_s = l1eq_pd(theta_0,A,[],y,1e-3);
xh = Psi1*theta_s;
```

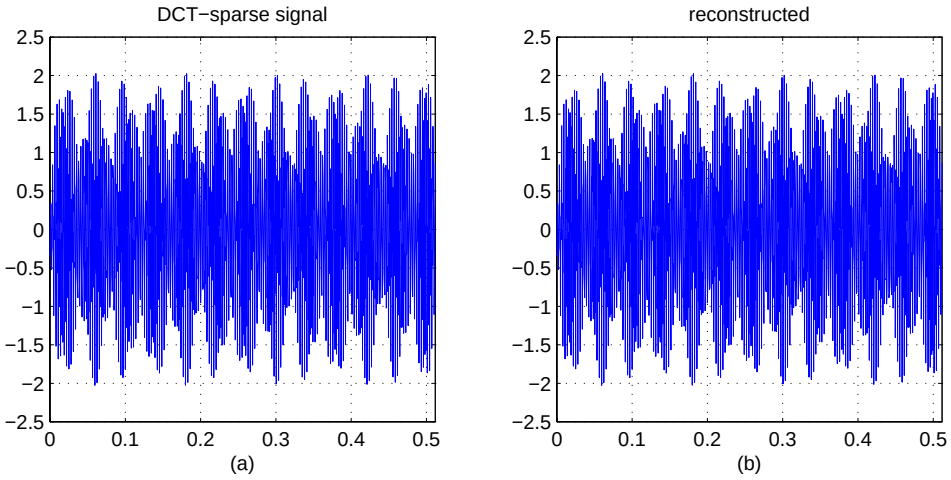


Figure 2.6

From Fig. 2.7, we see that with m = 300 (less than 4 times sparsity r) measurements, the recovery was practically perfect.

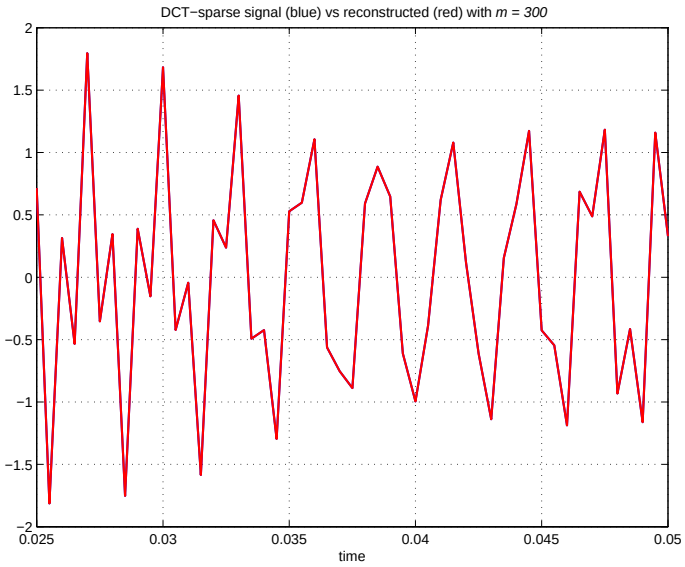


Figure 2.7

2.2 Compressed Sensing for Noisy Data

An important feature of compressed sensing is that it is robust in the sense that small perturbations in the data cause small deviations in the reconstruction. Here the "small perturbations in the data" refer either to signals that are not exactly sparse but nearly sparse, or to presence of noise in the sampling process. A signal model for CS that takes both of these issues into consideration is given by

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \tag{2.9a}$$

where w represents measurement noise and $\boldsymbol{A}$ is a sensing matrix of size m by n given by

$$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{\Phi}\boldsymbol{\Psi} = \hat{\boldsymbol{\Phi}}\boldsymbol{\Psi} \tag{2.9b}$$

In (2.9b), $\boldsymbol{S}$ is a matrix of size m by n that selects m rows from matrix, thus $\boldsymbol{S\Phi} = \hat{\boldsymbol{\Phi}}$, thus $\hat{\boldsymbol{\Phi}}$ is the same matrix introduced in (2.2). Note that for notion convenience we have used $\boldsymbol{x}$ here to denote the signal vector that was denoted by $\boldsymbol{\theta}$ in the noise-free counterpart of the above model (see model (2.2) with (2.1)).

As expected, signal $\boldsymbol{x}$ in (2.9) may be estimated from noisy measurement $\boldsymbol{y}$ by solving the convex minimization problem, called second-order cone program (SOCP), as follows.

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{x}\|_1 \\ &\text{subject to:} && \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon \end{aligned} \tag{2.10}$$

where ε is a bound of the amount of noise in the data. For notation convenience, we rephrase the l_1 minimization problem for noise free case as

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{x}\|_1 \\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{aligned} \tag{2.11}$$

The robustness of the CS relies heavily on a notion called *restricted isometry property* (RIP) and its connection to conditions similar to that in (2.3). RIP was first introduced and studied in [8].

Definition

For each integer r = 1, 2, …, define isometry constant δ_r of a matrix $\boldsymbol{A}$ as the smallest number such that

$$(1-\delta_r)\|\boldsymbol{x}\|_2^2 \le \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \le (1+\delta_r)\|\boldsymbol{x}\|_2^2 \tag{2.12}$$

for all r-sparse vectors $\boldsymbol{x}$. We shall say matrix $\boldsymbol{A}$ obeys the RIP of order r if δ_r in (2.12) is (less than one and) not too close to one. We shall make several intuitive observations on RIP before presenting the main results.

(i) (We know $\boldsymbol{A}$ cannot be) but just suppose $\boldsymbol{A}$ is an orthogonal matrix, then (2.12) holds for any vector x and any value of δ_r. Now, for an r-sparse $\boldsymbol{x}$, $\boldsymbol{A}\boldsymbol{x}$ is equal to $\boldsymbol{A}_r\boldsymbol{x}_r$ where $\boldsymbol{A}_r$ is a sub-matrix of $\boldsymbol{A}$ consisting of those r columns of $\boldsymbol{A}$ corresponding the r nonzero entries of $\boldsymbol{x}$, and $\boldsymbol{x}_r$ is vector $\boldsymbol{x}$ with its zero entries dropped (note that $\|\boldsymbol{x}\|_2 = \|\boldsymbol{x}_r\|_2$). Therefore, condition (2.12) is the same as saying that all sub-matrices of r columns taken from $\boldsymbol{A}$ are nearly orthogonal.

(ii) Obviously, condition (2.12) prevents any r-sparse $\boldsymbol{x}$ from being in the null space of $\boldsymbol{A}$. This is important in the context of CS, because if there exists an r-space $\boldsymbol{x}$ that belongs to the null space of $\boldsymbol{A}$ then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} = \boldsymbol{w}$ which means that the measurement obtained is totally useless as $\boldsymbol{y}$ contains no information whatsoever about that sparse signal $\boldsymbol{x}$.

(iii) In case δ_{2r} is sufficiently less than one, then for all r-sparse vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{x}_1 - \boldsymbol{x}_2$ is at most $2r$-sparse, hence we have

$$(1-\delta_{2r})\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \le \|\boldsymbol{A}\boldsymbol{x}_1 - \boldsymbol{A}\boldsymbol{x}_2\|_2^2 \le (1+\delta_{2r})\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \tag{2.13}$$

Assuming the noise-free case, an immediate consequence of (2.13) is that the measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ of an r-sparse signal $\boldsymbol{x}$ completely characterizes the signal in the sense if measurements $\boldsymbol{A}\boldsymbol{x}_1$ and $\boldsymbol{A}\boldsymbol{x}_2$ are identical, then signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ themselves must be identical.

Theorem 2 [9]

For the noise-free case, assume that $\delta_{2r} < \sqrt{2} - 1$. Then the solution $\boldsymbol{x}^*$ to problem (2.11) obeys

$$\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 \le C_0 \cdot \|\boldsymbol{x} - \boldsymbol{x}_r\|_1 / \sqrt{r} \tag{2.14a}$$

and

$$\|\boldsymbol{x} - \boldsymbol{x}^*\|_1 \le C_0 \cdot \|\boldsymbol{x} - \boldsymbol{x}_r\|_1 \tag{2.14b}$$

for some constant C_0, where $\boldsymbol{x}_r$ is the vector $\boldsymbol{x}$ with all but the largest r components set to zero.

Remark

If $\boldsymbol{x}$ is r-sparse, then $\boldsymbol{x}_r = \boldsymbol{x}$, hence (2.14) implies $\boldsymbol{x}^* = \boldsymbol{x}$, that is, the recovery is perfect. But the power of Theorem 2 is that it deals with non-sparse signals as well. In particular, if $\boldsymbol{x}$ is not sparse but approximately r-sparse, $\|\boldsymbol{x} - \boldsymbol{x}_r\|_1$ is small and Theorem 2 says the solution $\boldsymbol{x}^*$ to (2.11) is expected to be a reconstruction of $\boldsymbol{x}$ with good accuracy.

Theorem 3 [9]

For the case of noisy measurements with model (2.9), assume that $\delta_{2r} < \sqrt{2} - 1$. Then the solution $\boldsymbol{x}^*$ to problem (2.10) obeys

$$\left\| \boldsymbol{x} - \boldsymbol{x}^* \right\|_2 \le C_0 \cdot \left\| \boldsymbol{x} - \boldsymbol{x}_r \right\|_1 / \sqrt{r} + C_1 \varepsilon \tag{2.15}$$

for some constants C_0 and C_1. (For instance if $\delta_{2r} = 1/4$, then $C_0 \le 5.5$ and $C_1 \le 6$.)

Remark

Again, if $\boldsymbol{x}$ is r-sparse, then $\boldsymbol{x}_r = \boldsymbol{x}$, and (2.15) implies that $\left\| \boldsymbol{x} - \boldsymbol{x}^* \right\|_2 \le C_1 \varepsilon$. In other words, for sparse signals the solution accuracy is determined by the noise level. Like Theorem 2, Theorem 3 also deals with non-sparse signals. When $\boldsymbol{x}$ is not sparse but approximately r-sparse, Theorem 3 says the accuracy of the solution $\boldsymbol{x}^*$ to (2.10) is determined by both the closeness of signal $\boldsymbol{x}$ to its r-sparse counterpart and the noise level.

Example 2.3

The signal used in this example was constructed by sampling the same analog signal

$$x(t) = \sin(697\pi t) + \sin(1975\pi t)$$

at 2000 Hz as in Example 2. But in the present case we do not spasify it as we did in Example 2, hence signal $\boldsymbol{x}$ are only approximately sparse. We then randomly sample $\boldsymbol{x}$ with m = 300, and reconstruct the signal in the same way as in Example 2. The result is shown in Fig. 2.8.

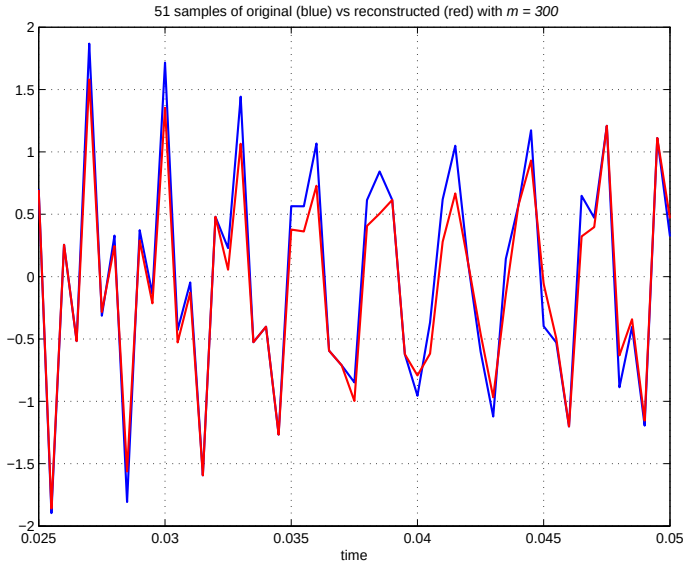


Figure 2.8

The reconstruction error can be increasing the number of measurements m. Fig. 2.9 shows the result with m = 380. In this sense the CS methodology possesses the desired scalability.

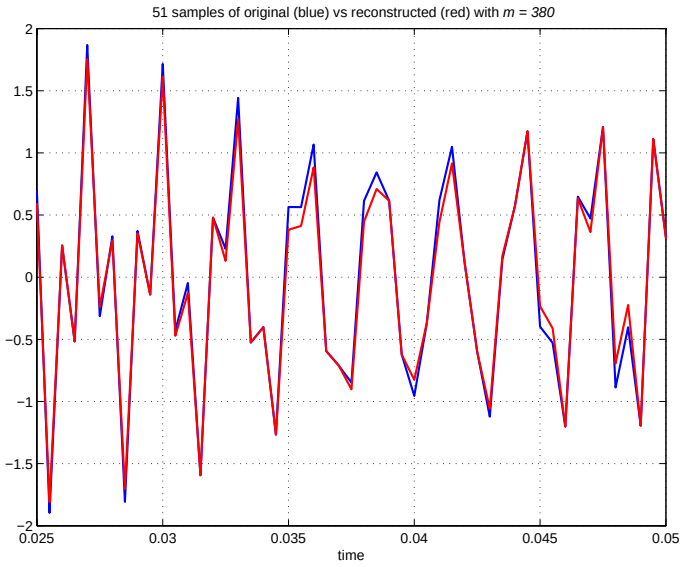


Figure 2.9

Next, a certain amount of noise was added to the measurements $\boldsymbol{y}$

```
randn('state',7)
w = 0.2*randn(m,1);
yn = y + w;
```

Figure 2.10 shows the difference between noise-free y and noise-contaminated $\boldsymbol{y}_n = \boldsymbol{y} + \boldsymbol{w}$.

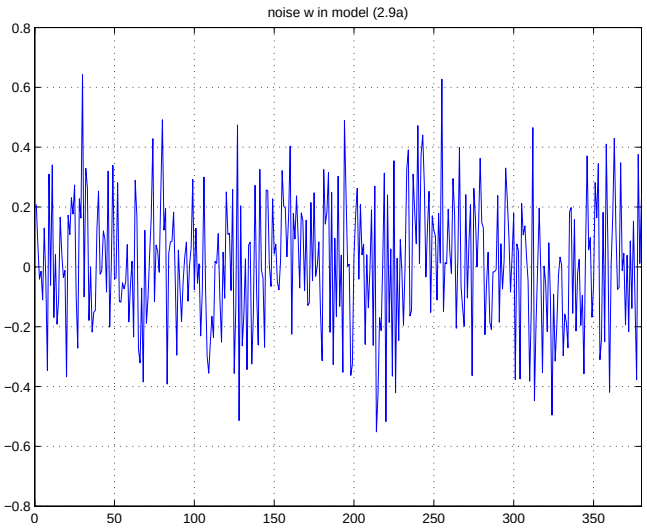


Figure 2.10

The Signal reconstruction using m = 380 noisy measurements was carried out by solving the SOCP problem (2.10) using l_1-MAGIC where the value of ε was set to $\varepsilon = 0.6\sqrt{m}\sigma$ with σ = 0.2 being the standard deviation of noise $\boldsymbol{w}$:

```
n = 1024;
m = 380;
sig = 0.2;      % standard deviation of white noise
dt = 1/2000;
```

```
T = 1023*dt;
t = 0:dt:T;
t = t(:);
x = sin(697*pi*t) + sin(1975*pi*t);
Dn = gen_dct(n);
% Get measurements
rand('state',15)
q = randperm(n);
q = q(:);
y = x(q(1:m));
randn('state',7)
w = sig*randn(m,1);
yn = y + w;
Psi1 = Dn';
% Reconstruction of signal x by SOCP
A = Psi1(q(1:m),:);
epsi = 0.6*sqrt(m)*sig;
theta_0 = A'*yn;
theta_s = l1qc_logbarrier(theta_0,A,[],yn,epsi);
xh = Psi1*theta_s;
```

The 51 samples of the reconstructed signal $\hat{\boldsymbol{x}}$ versus the original x over is depicted in Fig. 2.11.

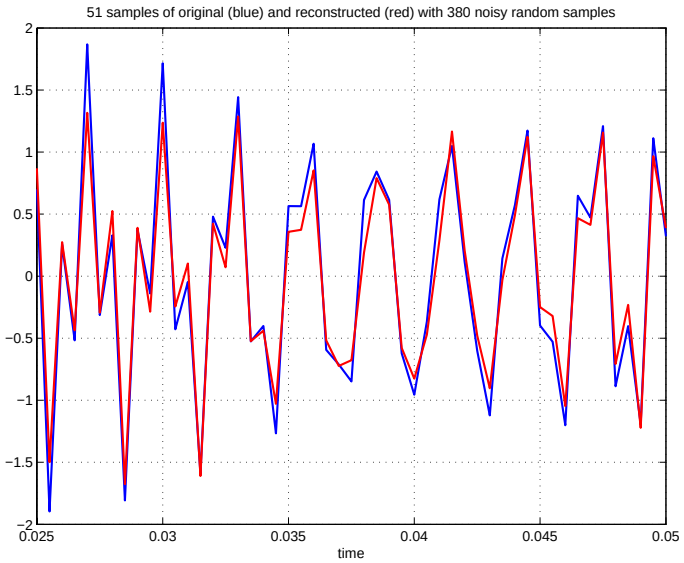


Figure 2.11

Example 2.4

In this example, we try to reconstruct the image letter "A" (see Fig. 1.9(a)) which contains $n = 2048$ pixels by using $m = 1000$ measurements through matrix $\boldsymbol{\Phi}$ which consists of the first m rows of DWT matrix W = gen_wave(2048,8) that involves Daubechies orthogonal wavelet D8: $\boldsymbol{\Phi}$ = W(1:m,:). The measurements $\boldsymbol{y}$ are collected using $\boldsymbol{y} = \boldsymbol{\Phi x}$ where x = A(:). Differing from l_1 minimization for 1-D signals, the reconstruction here is carried out by minimizing the total variation (TV) of the image

subject to the measurement constraints $\boldsymbol{\Phi x} = \boldsymbol{y}$ (see Sec. 3.1 below for details). Again the toolbox l_1-MAGIC was used for the TV minimization. The reconstruction result is shown in Fig. 2.12.

```
x = A(:);
n = length(x);
m = 1000;
Wn = gen_wave(n,8);
Phi = Wn(1:m,:);
y = Phi*x;
x0 = Phi'*y;
xh = l1eq_pd(x0,Phi,[],y,1e-3,itrs); % itrs: max # of iterations
Ah = vec2mat(xh,64)';
```

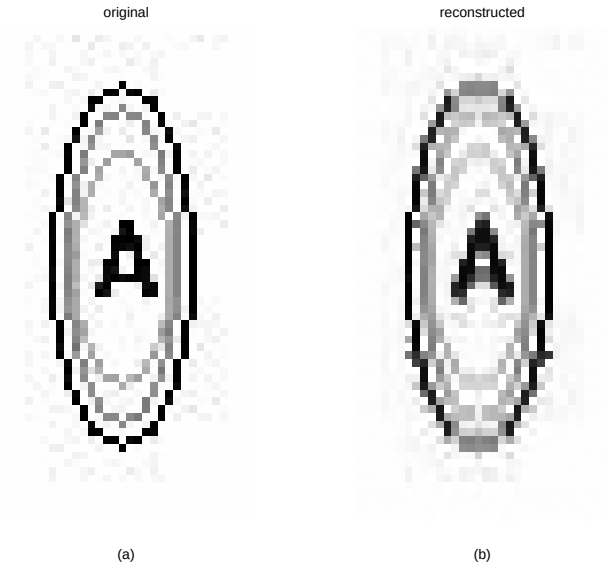


Figure 2. 12

III. Some Inverse Problems in Image Processing

3.1 Proximal-Gradient Methods for Convex Optimization Problems

A. Introduction

The purpose of this section is to introduce proximal-gradient methods for convex minimization problems [10] that are particularly suitable for inverse problems involving large-scale data sets. Specifically we are concerned with a class of unconstrained convex optimization problems

$$\text{minimize} \quad F(\boldsymbol{x}) \tag{3.1a}$$

where the objective function $F(x)$ is a *composite* type convex function

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}) \tag{3.1b}$$

with $g(\boldsymbol{x})$ a continuous convex function, possibly *nonsmooth*, and $f(\boldsymbol{x})$ a smooth convex function belonging to class $C^{1,1}$, i.e., functions that are continuously differentiable with *Lipschitz gradient*, namely,

$$\left\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\right\|_2 \le L(f)\left\|\boldsymbol{x}-\boldsymbol{y}\right\| \quad \text{for every } \boldsymbol{x},\boldsymbol{y} \tag{3.2}$$

where $L(f) > 0$ is a (Lipschitz) constant. Below is a "side note" on convex functions in class $C^{1,1}$.

▌ A smooth function $f(\boldsymbol{x})$ is convex if and only if on tangent lines of the function are underneath the function's graph. This statement can be written as

$$f(\boldsymbol{x}) \ge f(\boldsymbol{y}) + \left\langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle$$

If $f(\boldsymbol{x})$ is convex with Lipschitz gradient, then $f(\boldsymbol{x})$ is also bound from above by a family of local convex quadratic functions with Hessain $= L(f)\cdot \boldsymbol{I}$ [11]:

$$f(\boldsymbol{y}) + \left\langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle \le f(\boldsymbol{x}) \le f(\boldsymbol{y}) + \left\langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle + \frac{L(f)}{2}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|_2^2 \tag{3.3}$$

A proof of (3.3) can be found in [11] and is repeated below for convenience.
Proof:

$$f(\boldsymbol{x}) = f(\boldsymbol{y}) + \int_0^1 \left\langle \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})), \boldsymbol{x}-\boldsymbol{y} \right\rangle d\tau$$

$$= f(\boldsymbol{y}) + \left\langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle + \int_0^1 \left\langle \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle d\tau$$

Hence

$$\begin{aligned}
&\left| f(\boldsymbol{x}) - f(\boldsymbol{y}) - \left\langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle \right| \\
&= \left| \int_0^1 \left\langle \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle d\tau \right| \\
&\le \int_0^1 \left| \left\langle \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y} \right\rangle \right| d\tau \\
&\le \int_0^1 \left\| \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y}) \right\|_2 \cdot \left\| \boldsymbol{x}-\boldsymbol{y} \right\|_2 d\tau \\
&\le \int_0^1 \tau L(f) \left\| \boldsymbol{x}-\boldsymbol{y} \right\|_2^2 d\tau = \frac{L(f)}{2}\left\| \boldsymbol{x}-\boldsymbol{y} \right\|_2^2
\end{aligned}$$

The proof is complete. ▌

Examples

(i) The constrained convex problem

$$\begin{aligned} &\text{minimize} \quad h_0(\boldsymbol{x}) \\ &\text{subject to:} \quad \boldsymbol{x} \in C \end{aligned} \tag{3.4}$$

where $h_0(\boldsymbol{x})$ is a convex (possibly nonsmooth) function and C is a convex set in R^n. If we define an indicator function $\delta_C(\boldsymbol{x})$ as

$$\delta_C(\boldsymbol{x}) = \begin{cases} 0 & \text{for } \boldsymbol{x} \in C \\ +\infty & \text{for } \boldsymbol{x} \notin C \end{cases} \tag{3.5}$$

then (3.5) is equivalent to problem (3.1) with $f(\boldsymbol{x}) = 0$ and $g(\boldsymbol{x}) = h_0(\boldsymbol{x}) + \delta_C(\boldsymbol{x})$.

(ii) To address problem (2.10), namely

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{x}\|_1 \\ &\text{subject to:} && \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon \end{aligned}$$

it is often found more natural to study the closely related problem

$$\text{minimize} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \tag{3.6}$$

where parameter $\lambda > 0$ acts as a tradeoff between the sparsity of $\boldsymbol{x}$ and approximation error. Obviously, problem (3.6) is equivalent to problem (3.1) with $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ and $g(\boldsymbol{x}) = \lambda \|\boldsymbol{x}\|_1$.

(iii) Model (3.1) also covers convex formulations where the design variable is a *matrix* rather than a vector that provides a great deal of convenience for problems in which the objects of interest are two-dimensional such as digital still images. A popular example of such convex formulation is the total variation (TV) minimization of image subject to constraint on image fidelity to observed data [12]. The image model in this case is very simple:

$$\boldsymbol{x} + \boldsymbol{w} = \boldsymbol{b} \tag{3.7}$$

where $\boldsymbol{b}$ is the observed image and $\boldsymbol{w}$ is noise, and all items involved are matrices. Reference [10] was the first to suggest to estimate image $\boldsymbol{x}$ based on data $\boldsymbol{b}$ by solving the convex problem

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{x}\|_{TV} \\ &\text{subject to:} && \|\boldsymbol{x} - \boldsymbol{b}\|_F \le \varepsilon \end{aligned} \tag{3.8}$$

▌ As a side note, the TV norm of a matrix (a discretized image) $\boldsymbol{x}$ of size n_1 by n_2 is defined as

$$\|\boldsymbol{x}\|_{TV} = \sum_{i=1}^{n_1-1} \sum_{j=1}^{n_2-1} \|\boldsymbol{D}_{ij}\boldsymbol{x}\|_2, \quad \boldsymbol{D}_{ij}\boldsymbol{x} = \begin{bmatrix} x(i+1, j) - x(i, j) \\ x(i, j+1) - x(i, j) \end{bmatrix} \tag{3.9}$$

which is a discrete counterpart of the definition of total variation for a bounded variation (BV) function $u(x_1,x_2)$ defined over a region Ω:

$$TV[u(x_1, x_2)] = \iint_{\Omega} \sqrt{\left(\frac{\partial u}{\partial x_1}\right)^2 + \left(\frac{\partial u}{\partial x_2}\right)^2}\, dx_1 dx_2 \quad \blacksquare$$

Like in example (ii), (3.8) can be converted to an unconstrained convex problem

$$\text{minimize} \quad \|\boldsymbol{x} - \boldsymbol{b}\|_F^2 + \lambda \|\boldsymbol{x}\|_{TV} \tag{3.10}$$

where $\|\bullet\|_F$ denotes the Frobenius norm of the matrix. Clearly problem (3.10) fits nicely into (3.1) with $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{b}\|_F^2$ and $g(\boldsymbol{x}) = \lambda \|\boldsymbol{x}\|_{TV}$.

(iv) Yet another example, model (3.1) also covers convex problem

$$\text{minimize} \quad \|\mathcal{A}\boldsymbol{x}-\boldsymbol{b}\|_F^2+\lambda\|\boldsymbol{x}\|_{TV} \tag{3.11}$$

where $\boldsymbol{x}$ is a matrix variable and $\mathcal{A}$ represents a linear blurring operator. For blurring due to uniform linear motion or Gaussian lowpss filtering, $\mathcal{A}$ corresponds to a convolutional linear operator characterized by a small-size kernel. As a result, the first term in (3.11) remains smooth and convex, thus can be taken as function $f(\boldsymbol{x})$ in (3.1). As in example (iii), the second term of (3.11) is taken as function $g(\boldsymbol{x})$ in (3.1).

B. A Proximal-Gradient Algorithm for Problem (3.1)

We begin by considering a simple problem of minimizing function $f(\boldsymbol{x})$ which as in (3.1) assumed to be smooth and convex with smooth Lipschitz gradient. In the kth iteration of a conventional gradient method, known as the steepest descent method, iterate $\boldsymbol{x}_{k-1}$ is update to

$$\boldsymbol{x}_k=\boldsymbol{x}_{k-1}-t_k\nabla f(\boldsymbol{x}_{k-1}) \tag{3.12}$$

where $t_k>0$ is a scalar known as step size. One can easily verify that iterate $\boldsymbol{x}_{k-1}$ specified by (3.12) may be interpreted as the solution of a simple quadratic problem:

$$\boldsymbol{x}_k=\arg\min_{\boldsymbol{x}}\left\{f(\boldsymbol{x}_{k-1})+\langle(\boldsymbol{x}-\boldsymbol{x}_{k-1}),\nabla f(\boldsymbol{x}_{k-1})\rangle+\frac{1}{2t_k}\|\boldsymbol{x}-\boldsymbol{x}_{k-1}\|_2^2\right\} \tag{3.13}$$

By neglecting constant terms, (3.13) can be written as

$$\boldsymbol{x}_k=\arg\min_{\boldsymbol{x}}\left\{\frac{1}{2t_k}\|\boldsymbol{x}-(\boldsymbol{x}_{k-1}-t_k\nabla f(\boldsymbol{x}_{k-1}))\|_2^2\right\} \tag{3.14}$$

By extending this gradient updating mechanism to problem (3.1), we obtain

$$\boldsymbol{x}_k=\arg\min_{\boldsymbol{x}}\left\{\frac{1}{2t_k}\|\boldsymbol{x}-(\boldsymbol{x}_{k-1}-t_k\nabla f(\boldsymbol{x}_{k-1}))\|_2^2+g(\boldsymbol{x})\right\} \tag{3.15}$$

Example 3.1 For the problem of compressed sensing of noisy signals formulated by (3.5), i.e.

$$\text{minimize} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{x}\|_1$$

(3.15) becomes

$$\boldsymbol{x}_k=\arg\min_{\boldsymbol{x}}\left\{\frac{1}{2t_k}\|\boldsymbol{x}-(\boldsymbol{x}_{k-1}-t_k\nabla f(\boldsymbol{x}_{k-1}))\|_2^2+\lambda\|\boldsymbol{x}\|_1\right\} \tag{3.16}$$

or equivalently,

$$\boldsymbol{x}_k=\arg\min_{\boldsymbol{x}}\left\{\frac{1}{2t_k}\|\boldsymbol{x}-\boldsymbol{c}_k\|_2^2+\lambda\|\boldsymbol{x}\|_1\right\} \tag{3.17a}$$

where

$$\boldsymbol{c}_k=\boldsymbol{x}_{k-1}-t_k\nabla f(\boldsymbol{x}_{k-1})=\boldsymbol{x}_{k-1}-2t_k\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_{k-1}-\boldsymbol{y}) \tag{3.17b}$$

Because both the 1-norm and square of the 2-norm are *separable*, i.e. each of them is mere the sum of *n nonnegative* terms and each of these terms involves only a *single* (scalar) variable, the iterate $\boldsymbol{x}_k$ in (3.17a) can be computed exactly by a straightforward *shrinkage* step (assuming $\boldsymbol{c}_k$ in (3.17b) has been calculated) as

$$\boldsymbol{x}_k = \mathcal{T}_{\lambda t_k}(\boldsymbol{c}_k) \tag{3.18}$$

where $\mathcal{T}_\alpha$ is an shrinkage operator which maps R^n to R^n with the ith entry of the output vector given by

$$\mathcal{T}_\alpha(\boldsymbol{c})\big|_i = \left(|c_i| - \alpha\right)_+ \operatorname{sgn}(c_i) \tag{3.19}$$

with $(u)_+ = \max(u, 0)$. For this reason the iterative algorithm based on (3.18) is known as one of the iterative shrinkage-thresholding algorithms (ISTAs) [13]-[16]. It can be shown that the algorithm converges to a solution of (3.6) if the step size is selected to satisfy $t_k \in \left(0,\ 1/\left\|\boldsymbol{A}^T\boldsymbol{A}\right\|\right)$. On the other hand, the convergence of ISTA type of algorithms has been known to be slow.

As a numerical example, ISTA was applied to the problem discussed in Example 2.3 with identical data set. In this case the interval for step size t_k was (0, 1) and in this example t_k was fixed to 0.95. With $\lambda = 0.35$ and 34 iterations, the algorithm converged to a solution whose 51 samples over time interval [0.025, 0.05] are shown in Fig. 3.1 in comparison with the original signal.

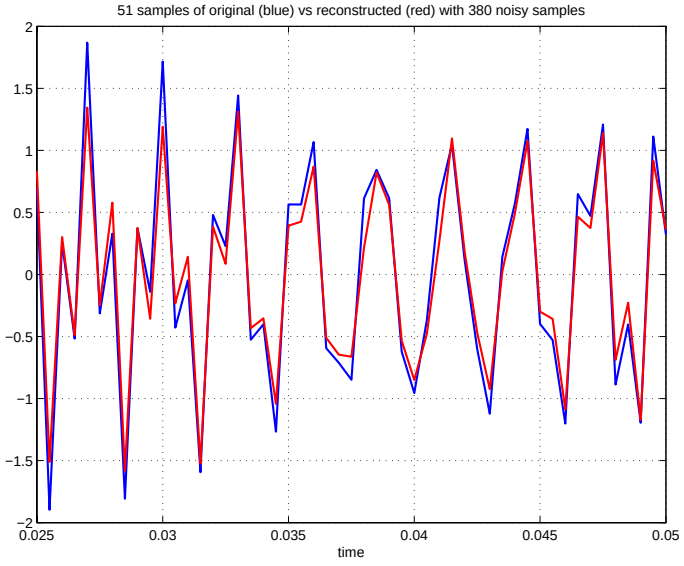


Figure 3.1

```
% Program: ex3_1.m
% To implement the ISTA algorithm to the problem in Example 3.1
% (see pap. 36 - 38 of the notes).
% Input:
% m: number of rows (measurements) in matrix A.
% sig: standard deviation of noise w.
% lam: parameter \lambda.
% itrs: number of iterations to be performed.
% st1: initial randon state for random permutation of index {1, 2, ..., n}
% st2: initial random state for measurement noise w.
% Output:
% x: original signal.
% xh: reconstructed signal.
% err: relative reconstruction error.
```

```
% Written by W.-S. Lu, University of Victoria.
% Example: [x,xh,err] = ex3_1(380,0.2,0.35,34,15,7);
function [x,xh,err] = ex3_1(m,sig,lam,itrs,st1,st2)
% Prepare a signal x who is sparse in DCT domain
n = 1024;
dt = 1/2000;
T = 1023*dt;
t = 0:dt:T;
t = t(:);
x = sin(697*pi*t) + sin(1975*pi*t);
Dn = gen_dct(n);
% Get measurements
rand('state',st1)
q = randperm(n);
q = q(:);
y = x(q(1:m));
randn('state',st2)
w = sig*randn(m,1);
yn = y + w;
Psi1 = Dn';
% Reconstruction of signal x by l1-minimization
A = Psi1(q(1:m),:);
tmax = 1/max(eig(A*A'));
tt = 0.95*tmax;
% thk = A'*yn;
thk = zeros(n,1);
alp = lam*tt;
for i = 1:itrs,
   ck = thk - 2*tt*A'*(A*thk-yn);
   thk = (max(abs(ck)-alp,0)).*sign(ck);
end
xh = Psi1*thk;
err = norm(x-xh)/norm(x);
figure(1)
subplot(121)
plot(t,x)
title('DCT-sparse signal')
xlabel('(a)')
axis([0 T -2.5 2.5])
grid
subplot(122)
plot(t,xh)
title('reconstructed')
xlabel('(b)')
axis([0 T -2.5 2.5])
grid
figure(2)
```

```
t1 = 50*dt:dt:100*dt;
plot(t1,x(50:100),'b',t1,xh(50:100),'r','linewidth', 1.5);
grid
xlabel('time')
title('51 samples of original (blue) vs reconstructed (red) with 380 noisy samples')
```

```
% Program: gen_dct.m
% To produce a matrix C of size n x n that performs
% n-point 1-D DCT for signal x by C*x. See example 1.1
% on p. 10 of the notes.
% Input:
% n: size of DCT matrix C.
% Output:
% C: DCT matrix.
% Written by W.-S. Lu, University of Victoria.
function C = gen_dct(n)
alp = [sqrt(1/n) sqrt(2/n)*ones(1,n-1)];
ind = (1:2:(2*n-1))*pi/(2*n);
C = zeros(n,n);
for i = 1:n,
    C(i,:) = alp(i)*cos((i-1)*ind);
end
```

In a recent work [15] by Beck and Teboulle, some early work of Nesterov [16] was extended in developing a fast iterative shrinkage-thresholding algorithm (FISTA) for general convex problem (3.1). The development in [16] starts with a model that is quite similar to that in (3.15), with t_k replaced by a constant $1/L$ which will be related to the Lipschitz constant $L(f)$ in (3.2) later. To get there, we approximate function $f(\boldsymbol{x}) + g(\boldsymbol{x})$ at point $\boldsymbol{x}_{k-1}$ by quadratic function

$$Q_L(\boldsymbol{x}, \boldsymbol{x}_{k-1}) = f(\boldsymbol{x}_{k-1}) + \langle (\boldsymbol{x} - \boldsymbol{x}_{k-1}), \nabla f(\boldsymbol{x}_{k-1}) \rangle + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_{k-1}\|_2^2 + g(\boldsymbol{x}) \tag{3.19}$$

whose unique minimizer is given by

$$p_L(\boldsymbol{x}_{k-1}) = \arg\min_{\boldsymbol{x}} \left\{ \frac{L}{2} \left\| \boldsymbol{x} - \left(\boldsymbol{x}_{k-1} - \frac{1}{L} \nabla f(\boldsymbol{x}_{k-1}) \right) \right\|_2^2 + g(\boldsymbol{x}) \right\} \tag{3.20}$$

(3.20) is the same as (3.14) if one replaces t_k there by $1/L$. In this way, the primary step of ISTA becomes

$$\boldsymbol{x}_k = P_L(\boldsymbol{x}_{k-1}) \tag{3.21}$$

with L set to $1/ t_k$. Note that as long as the constant L in (3.19) is taken to be no less than Lip Contant $L(f)$, it follows from (3.3) that

$$f(\boldsymbol{x}) + g(\boldsymbol{x}) \le f(\boldsymbol{x}_{k-1}) + \langle \nabla f(\boldsymbol{x}_{k-1}), \boldsymbol{x} - \boldsymbol{x}_{k-1} \rangle + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_{k-1}\|_2^2 + g(\boldsymbol{x}) \tag{3.22}$$

Note that the right-hand side of (3.22) is precisely equal to $Q_L(\boldsymbol{x}, \boldsymbol{x}_{k-1})$ in (3.19). In other words, $Q_L(\boldsymbol{x}, \boldsymbol{x}_{k-1})$ is an easier-to-deal-with convex upper bound of the objective function $F(\boldsymbol{x})$ and by minimizing the upper bound, $Q_L(\boldsymbol{x}_k, \boldsymbol{x}_{k-1})$ with $\boldsymbol{x}_k$ given by (3.21) offers a tight upper bound of $F(\boldsymbol{x})$, provided that $L \geq L(f)$.

In summary, ISTA for general composite objective function $F(\boldsymbol{x})$ in (3.1) is outlined as follows.

ISTA with Constant Step Size [15]

Input $L = L(f)$ – A Lipschitz constant of $\nabla f(\boldsymbol{x})$.

Step 0 Take $\boldsymbol{x}_0 \in R^n$.

Step k ($k \geq 1$) Compute $\boldsymbol{x}_k = P_L(\boldsymbol{x}_{k-1})$ by solving the problem in (3.20).

It is shown [15] that the sequence $\{\boldsymbol{x}_k\}$ generated by ISTA satisfies

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \frac{L(f)\left\|\boldsymbol{x}_0 - \boldsymbol{x}^*\right\|_2^2}{2k} \tag{3.23}$$

where $\boldsymbol{x}^*$ is a minimizer of problem (3.1).

C. Fast Iterative Shrinkage-Thresholding Algorithm for Problem (3.1)

Reference [15] also proposes a fast ISTA (FISTA) for problem (3.1), which is in spirit an extension of an algorithm by Nesterov [16]. The algorithm is outlined below.

FISTA with Constant Step Size [15]

Input $L = L(f)$ – A Lipschitz constant of $\nabla f(\boldsymbol{x})$.

Step 0 Take $\boldsymbol{y}_1 = \boldsymbol{x}_0 \in R^n$, $t_1 = 1$.

Step k ($k \geq 1$) Compute

(i) $\boldsymbol{x}_k = P_L(\boldsymbol{y}_k)$ by solving the problem in (3.20).

(ii) $t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2}$

(iii) $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \left(\dfrac{t_k - 1}{t_{k+1}}\right)(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$

It is shown [15] that the sequence $\{\boldsymbol{x}_k\}$ generated by FISTA satisfies

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \le \frac{2L(f)\left\|\boldsymbol{x}_0 - \boldsymbol{x}^*\right\|_2^2}{(k+1)^2} \tag{3.24}$$

where $\boldsymbol{x}^*$ is a minimizer of problem (3.1). On comparing the convergence rate of $O(k^{-1})$ for ISTA as evidenced in (3.23), FISTA enjoys a convergence rate of $O(k^{-2})$.

Example 3.2

As a numerical example, FISTA algorithm was applied to the problem discussed in Example 2.3 (as well as in Example 3.1) with identical data set. With $\lambda = 0.37$ and 40 iterations, the algorithm converges to a solution whose 51 samples over time interval [0.025, 0.05] are shown in Fig. 3.2 in comparison with the original signal. The reconstructed signal was found to be practically the same as that obtained from Example 3.1. Unfortunately, for reasons unknown to the lecturer, this example did not demonstrate the fast convergence property of FISTA.

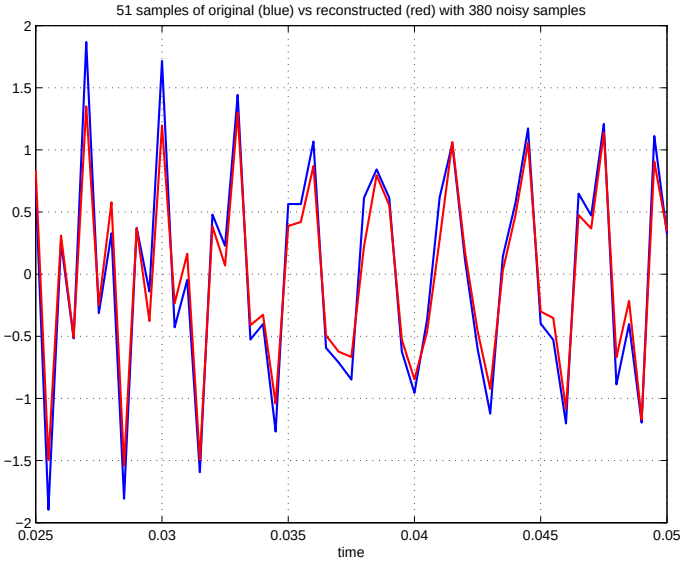


Figure 3.2

```
% Example: [x,xh,err] = ex3_2(380,0.2,0.37,40,15,7);
function [x,xh,err] = ex3_2(m,sig,lam,itrs,st1,st2)
% Prepare a signal x who is sparse in DCT domain
n = 1024;
dt = 1/2000;
T = 1023*dt;
t = 0:dt:T;
t = t(:);
x = sin(697*pi*t) + sin(1975*pi*t);
Dn = gen_dct(n);
% Get measurements
rand('state',st1)
q = randperm(n);
q = q(:);
y = x(q(1:m));
```

```
randn('state',st2)
w = sig*randn(m,1);
yn = y + w;
Psi1 = Dn';
% Reconstruction of signal x by l1-minimization
A = Psi1(q(1:m),:);
Lf = max(eig(A*A'));
Li = 1/Lf;
alp = lam*Li;
thk = zeros(n,1);
yk = thk;
tk = 1;
for i = 1:itrs,
   ck = yk - 2*Li*A'*(A*yk-yn);
   thk1 = (max(abs(ck)-alp,0)).*sign(ck);
   tk1 = 0.5 + 0.5*sqrt(1+4*tk^2);
   tt = (tk-1)/tk1;
   yk = thk1 -tt*(thk1-thk);
   tk = tk1;
   thk = thk1;
end
xh = Psi1*thk1;
err = norm(x-xh)/norm(x)
figure(1)
subplot(121)
plot(t,x)
title('DCT-sparse signal')
xlabel('(a)')
axis([0 T -2.5 2.5])
grid
subplot(122)
plot(t,xh)
title('reconstructed')
xlabel('(b)')
axis([0 T -2.5 2.5])
grid
figure(2)
t1 = 50*dt:dt:100*dt;
plot(t1,x(50:100),'b',t1,xh(50:100),'r','linewidth', 1.5);
grid
xlabel('time')
title('51 samples of original (blue) vs reconstructed (red) with 380 noisy samples')
```

3.2 Fast Gradient Projection Algorithms for Constrained TV-Based Image De-Noising

In this section we introduce a proximal-gradient algorithm proposed in [17] for image de-noising. In

Sec. 3.1, image de-noising problem was formulated in (3.10) as minimize $\|\boldsymbol{x}-\boldsymbol{b}\|_F^2 + \lambda\|\boldsymbol{x}\|_{TV}$ which is merely an instance of the general convex problem (3.1). In [17] the image de-noising problem is formulated as a constrained convex problem

$$\begin{aligned} &\text{minimize} && \|\boldsymbol{x}-\boldsymbol{b}\|_F^2 + 2\lambda\|\boldsymbol{x}\|_{TV} \\ &\text{subject to:} && \boldsymbol{x} \in B_{l,u} \equiv \{x_{i,j},\, l \le x_{ij} \le u\} \end{aligned} \tag{3.25}$$

where the bounds are imposed to reflect the fact the entries are the light intensity of image pixels, hence their values typically fall within, say, the interval of [0, 255]. The basic idea of [17] is to construct a dual problem of (3.25) in a way similar to that of [18] where the unconstrained problem of minimizing the objective function in (3.25) without the box constraints was solved by a gradient-based algorithm.

We begin by defining the TV norm of an image that is slightly different from (3.9). For image $\boldsymbol{x}$ of size n_1 by n_2, the discrete isotropic TV, denoted by $TV_I(\boldsymbol{x})$, and anisotropic TV, denoted by $TV_{l1}(\boldsymbol{x})$, are defined, respectively, as

$$\begin{aligned} TV_I(\boldsymbol{x}) = &\sum_{i=1}^{n_1-1}\sum_{j=1}^{n_2-1}\sqrt{\left(x_{i,j}-x_{i+1,j}\right)^2+\left(x_{i,j}-x_{i,j+1}\right)^2} \\ &+\sum_{i=1}^{n_1-1}\left|x_{i,n_2}-x_{i+1,n_2}\right|+\sum_{j=1}^{n_2-1}\left|x_{n_1,j}-x_{n_1,j+1}\right| \end{aligned} \tag{3.26}$$

and

$$\begin{aligned} TV_{l_1}(\boldsymbol{x}) = &\sum_{i=1}^{n_1-1}\sum_{j=1}^{n_2-1}\left\{\left|x_{i,j}-x_{i+1,j}\right|+\left|x_{i,j}-x_{i,j+1}\right|\right\} \\ &+\sum_{i=1}^{n_1-1}\left|x_{i,n_2}-x_{i+1,n_2}\right|+\sum_{j=1}^{n_2-1}\left|x_{n_1,j}-x_{n_1,j+1}\right| \end{aligned} \tag{3.27}$$

To describe the algorithm, we need to define four more items.

- Let $\mathcal{P}$ be the set of matrix pairs $(\boldsymbol{p}, \boldsymbol{q})$ where $\boldsymbol{p} \in R^{(n_1-1)\times n_2}$, $\boldsymbol{q} \in R^{n_1\times(n_2-1)}$ satisfy

$$\begin{aligned} &p_{i,j}^2 + q_{i,j}^2 \le 1, \quad \text{for } 1 \le i \le n_1-1,\ 1 \le j \le n_2-1 \\ &\left|p_{i,n_2}\right| \le 1, \quad \text{for } 1 \le i \le n_1-1 \\ &\left|q_{n_1,j}\right| \le 1, \quad \text{for } 1 \le j \le n_2-1 \end{aligned} \tag{3.28}$$

- Define linear operator $\mathcal{L}$ that maps an element $(\boldsymbol{p}, \boldsymbol{q})$ of $\mathcal{P}$ to an n_1-by-n_2 matrix as

$$\left(\mathcal{L}(\boldsymbol{p},\boldsymbol{q})\right)_{i,j} = p_{i,j} - p_{i-1,j} + q_{i,j} - q_{i,j-1} \quad \text{for } 1 \le i \le n_1,\ 1 \le j \le n_2 \tag{3.29}$$

where $p_{0,j} = p_{n1,j} = q_{i,0} = q_{i,n2} = 0$ for $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$.

- The adjoint of operator $\mathcal{L}$, denoted by $\mathcal{L}^T$, is a linear map from an n_1-by-n_2 matrix $\boldsymbol{x}$ to a matrix pair $(\boldsymbol{p}, \boldsymbol{q})$ of $\mathcal{P}$, and is defined by

$$\mathcal{L}^T(\boldsymbol{x}) = (\boldsymbol{p}, \boldsymbol{q}) \tag{3.30a}$$

where

$$\begin{aligned} p_{i,j} &= x_{i,j} - x_{i+1,j}, \quad \text{for } 1 \le i \le n_1 - 1, \ 1 \le j \le n_2 \\ q_{i,j} &= x_{i,j} - x_{i,j+1}, \quad \text{for } 1 \le i \le n_1, \ 1 \le j \le n_2 - 1 \end{aligned} \tag{3.30b}$$

• Finally, we denote the orthogonal projection operator onto set C by $\boldsymbol{P}_C$. In particular, if C is the set $B_{l,u}$ of the box constraints in (3.25), then we have

$$\left(\boldsymbol{P}_{B_{l,u}}(\boldsymbol{x})\right)_{i,j} = \begin{cases} l & \text{if } x_{i,j} < l \\ x_{i,j} & \text{if } l \le x_{i,j} \le u \\ u & \text{if } x_{i,j} > u \end{cases} \tag{3.31}$$

One of the main results of [17] is the following theorem.

Theorem 4 [17]

Let $(\boldsymbol{p}, \boldsymbol{q})$ be the solution of the dual problem

$$\underset{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}{\text{minimize}} \quad h(\boldsymbol{p},\boldsymbol{q}) \equiv -\left\|\boldsymbol{H}_{B_{l,u}}(\boldsymbol{b} - \lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q}))\right\|_F^2 + \left\|\boldsymbol{b} - \lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})\right\|_F^2 \tag{3.32a}$$

where

$$\boldsymbol{H}_{B_{l,u}}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{P}_{B_{l,u}}(\boldsymbol{x}) \tag{3.32b}$$

Then the solution of problem (3.25) with TV = TV_I is given by

$$\boldsymbol{x} = \boldsymbol{P}_{B_{l,u}}\left(b - \lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})\right) \tag{3.33}$$

Proof [17]:

By a property of norm, we can write

$$\begin{aligned} \sqrt{x^2 + y^2} &= \max_{p_1,p_2}\left\{p_1 x + p_2 y : p_1^2 + p_2^2 \le 1\right\} \\ |x| &= \max_{p}\left\{px : |p| \le 1\right\} \end{aligned} \tag{3.34}$$

It follows that the isotropic TV norm TV_I can be expressed as

$$TV_I(\boldsymbol{x}) = \underset{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}{\text{maximize}} T(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}) \tag{3.35}$$

where

$$T(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{n_1-1}\sum_{j=1}^{n_2-1}\left[p_{i,j}\left(x_{i,j} - x_{i+1,j}\right) + q_{i,j}\left(x_{i,j} - x_{i,j+1}\right)\right] + \sum_{i=1}^{n_1-1} p_{i,n_2}(x_{i,n_2} - x_{i+1,n_2}) + \sum_{j=1}^{n_2-1} q_{n_1,j}(x_{n_1,j} - x_{n_1,j+1}) \tag{3.36}$$

which implies that

$$T(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}) = \text{Tr}\left(\mathcal{L}(\boldsymbol{p},\boldsymbol{q})^T \boldsymbol{x}\right) \tag{3.37}$$

Hence problem (3.25) can be formulated as

$$\underset{\boldsymbol{x}\in B_{l,u}}{\text{minimize}}\ \underset{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}{\text{maximize}}\left\{\|\boldsymbol{x}-\boldsymbol{b}\|_F^2+2\lambda\text{Tr}\left[\mathcal{L}(\boldsymbol{p},\boldsymbol{q})^T\boldsymbol{x}\right]\right\} \tag{3.38}$$

Because the objective unction in (3.38) in convex in $\boldsymbol{x}$ and concave in ($\boldsymbol{p}$, $\boldsymbol{q}$), it is allowed to change the order of the minimization and maximization (see e.g. [19, Corollary 37.3.2]) and the problem at hand becomes

$$\underset{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}{\text{maximize}}\ \underset{\boldsymbol{x}\in B_{l,u}}{\text{minimize}}\left\{\|\boldsymbol{x}-\boldsymbol{b}\|_F^2+2\lambda\text{Tr}\left[\mathcal{L}(\boldsymbol{p},\boldsymbol{q})^T\boldsymbol{x}\right]\right\} \tag{3.39}$$

which can be written as

$$\max_{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}\min_{\boldsymbol{x}\in B_{l,u}}\left\{\|\boldsymbol{x}-(\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q}))\|_F^2-\|\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})\|_F^2+\|\boldsymbol{b}\|_F^2\right\} \tag{3.40}$$

The solution of the inner minimization problem in (3.40) is given by

$$\boldsymbol{x}=\boldsymbol{P}_{B_{l,u}}(\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})) \tag{3.41}$$

where ($\boldsymbol{p}$, $\boldsymbol{q}$) is the solution of problem (3.40) when one substitutes (3.41) back into (3.40):

$$\max_{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{\mathcal{P}}}\left\{\left\|\boldsymbol{P}_{B_{l,u}}(\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q}))-(\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q}))\right\|_F^2-\|\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})\|_F^2\right\}$$

which is the same dual problem (3.32), hence the proof is complete. ■

It can be shown that function $h(\boldsymbol{p}, \boldsymbol{q})$ in (3.32a) is continuously differentiable and its gradient is given by [17]

$$\nabla h(\boldsymbol{p},\boldsymbol{q})=-2\lambda\mathcal{L}^T\boldsymbol{P}_{B_{l,u}}(\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p},\boldsymbol{q})) \tag{3.42}$$

Based on Theorem 4, formula (3.42) and a few lines of further analysis (see [17] for details), a gradient projection (GP) algorithm is proposed in [17] for solving problem (3.25). The algorithm is outlined below.

Algorithm GP($\boldsymbol{b}$, λ, K)

Inputs: observed image $\boldsymbol{b}$, regularization parameter λ, and iteration number K.

Output: An optimal solution $\boldsymbol{x}^*$ of (3.25) up to a tolerance.

Step 0 Take $(\boldsymbol{p}_0,\boldsymbol{q}_0)=(\boldsymbol{O}_{(n_1-1)\times n_2},\boldsymbol{O}_{n_1\times(n_2-1)})$.

Step k (k = 1, 2, …, K) Compute

$$(\boldsymbol{p}_k,\boldsymbol{q}_k)=\boldsymbol{P}_{\boldsymbol{\mathcal{P}}}\left\{(\boldsymbol{p}_{k-1},\boldsymbol{q}_{k-1})+\frac{1}{8\lambda}\mathcal{L}^T\left(\boldsymbol{P}_{B_{l,u}}\left[\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p}_{k-1},\boldsymbol{q}_{k-1})\right]\right)\right\} \tag{3.43}$$

Set $\boldsymbol{x}^*=\boldsymbol{P}_{B_{l,u}}\left[\boldsymbol{b}-\lambda\mathcal{L}(\boldsymbol{p}_K,\boldsymbol{q}_K)\right]$.

The projection operator $\boldsymbol{P}_{\boldsymbol{\mathcal{P}}}$ in (3.43), which maps a matrix pair ($\boldsymbol{p}$, $\boldsymbol{q}$) to another matrix pair ($\boldsymbol{r}$, $\boldsymbol{s}$) = $\boldsymbol{P}_{\boldsymbol{\mathcal{P}}}(\boldsymbol{p}, \boldsymbol{q})$, can be readily implemented as follows:

$$r_{i,j} = \begin{cases} \dfrac{p_{i,j}}{\max\left\{1, \sqrt{p_{i,j}^2 + q_{i,j}^2}\right\}} & i = 1, \ldots, n_1 - 1, \;\; j = 1, \ldots, n_2 - 1 \\ \dfrac{p_{i,n_2}}{\max\left\{1, \left| p_{i,n_2} \right|\right\}} & i = 1, \ldots, n_1 - 1 \end{cases}$$

$$s_{i,j} = \begin{cases} \dfrac{q_{i,j}}{\max\left\{1, \sqrt{p_{i,j}^2 + q_{i,j}^2}\right\}} & i = 1, \ldots, n_1 - 1, \;\; j = 1, \ldots, n_2 - 1 \\ \dfrac{q_{n_1,j}}{\max\left\{1, \left| q_{n_1,j} \right|\right\}} & j = 1, \ldots, n_2 - 1 \end{cases} \tag{3.44}$$

A fast gradient projection (FGP) algorithm is also proposed in [17], which in spirit minims the extension of ISTA to FISTA. The algorithm steps are sketched as follows.

Algorithm FGP($\boldsymbol{b}, \lambda$, K)
Inputs: observed image $\boldsymbol{b}$, regularization parameter λ, and iteration number K.

Output: An optimal solution $\boldsymbol{x}^*$ of (3.25) up to a tolerance.

Step 0 Take $(\boldsymbol{r}_1, \boldsymbol{s}_1) = (\boldsymbol{p}_0, \boldsymbol{q}_0) = (\mathbf{O}_{(n_1-1)\times n_2}, \mathbf{O}_{n_1\times(n_2-1)})$, $t_1 = 1$.

Step k (k = 1, 2, …, K)
Compute

$$(\boldsymbol{p}_k, \boldsymbol{q}_k) = \boldsymbol{P}_{\mathcal{P}}\left\{(\boldsymbol{r}_k, \boldsymbol{s}_k) + \frac{1}{8\lambda}\mathcal{L}^T\left(\boldsymbol{P}_{B_{l,u}}\left[\boldsymbol{b} - \lambda\mathcal{L}(\boldsymbol{r}_k, \boldsymbol{s}_k)\right]\right)\right\} \tag{3.45a}$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \tag{3.45b}$$

$$(\boldsymbol{r}_{k+1}, \boldsymbol{s}_{k+1}) = (\boldsymbol{p}_k, \boldsymbol{q}_k) + \left(\frac{t_k - 1}{t_{k+1}}\right)(\boldsymbol{p}_k - \boldsymbol{p}_{k-1}, \boldsymbol{q}_k - \boldsymbol{q}_{k-1}) \tag{3.45c}$$

Set $$\boldsymbol{x}^* = \boldsymbol{P}_{B_{l,u}}\left[\boldsymbol{b} - \lambda\mathcal{L}(\boldsymbol{p}_K, \boldsymbol{q}_K)\right] \tag{3.46}$$

Example 3.2 The GP algorithm was applied to image cameraman of size 256 by 256 which was corrupted with while noise with standard deviation 0.1. With λ = 0.1 and K = 300, the GP algorithm produced an image whose PSNR was improved from 20 dB to 27.5051 dB, see Fig. 3.3.

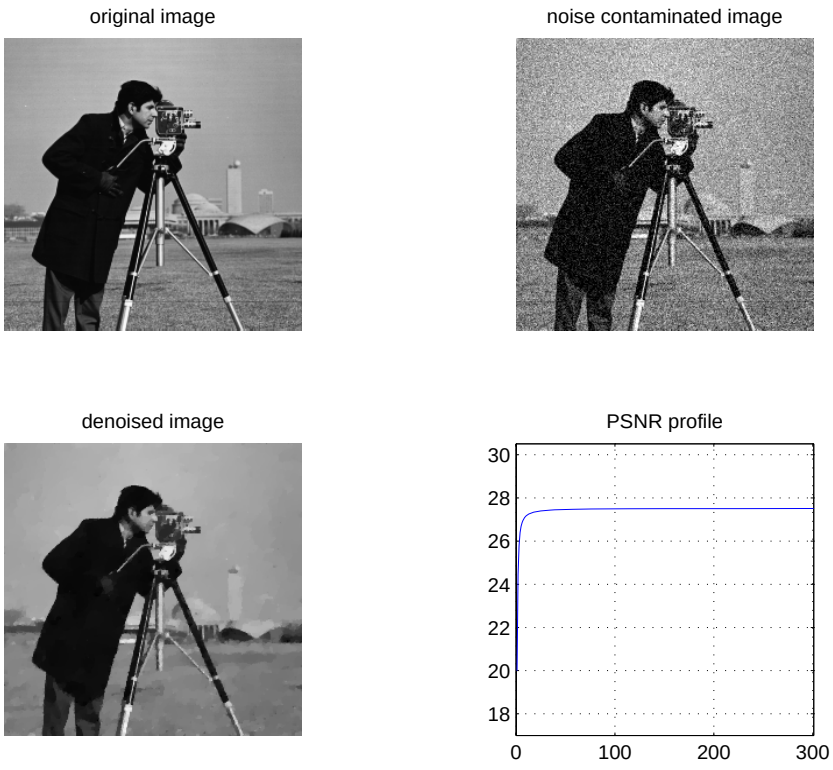


Figure 3.3

```
% Program: gp_denoise.m
% To remove noise from a corrupted still image by using
% the gradient projection (GP) algorithm described in Sec. 3.2
% of the notes, see pp. 41-44.
% Reference: A. Beck and M. Teboulle, "Fast gradient-based
% algorithms for constrained total variation image denoising
% and deblurring problems," IEEE Trans. Image Processing,
% vol. 18, no. 11, pp. 2419-2434, Nov. 2009.
% Input:
% xorig: original image.
% st: initial random state for noise.
% sig: standard deviation of the noise.
% lam: parameter \lambda.
% K: number of iterations to be performed.
% Output:
% x: de-noised image.
% psnr: profile of peak signal-to-noise ratio in dB.
% Written by W.-S. Lu, University of Victoria.
% Example: [x,psnr] = gp_denoise(camera256,17,0.1,0.1,300);
function [x,psnr] = gp_denoise(xorig,st,sig,lam,K)
xorig = xorig/255;
[m,n] = size(xorig);
pk = zeros(m-1,n);
qk = zeros(m,n-1);
randn('state',st)
```

```
u = sig*randn(m,n);
xin = xorig + u;
k = 0;
Li = 1/(8*lam);
c = sqrt(m*n);
psnr_before = 20*log10(c/norm(u,'fro'));
psnr = zeros(K+1,1);
psnr(1) = psnr_before;
while k < K,
      xpq = proj_bound(xin - lam*oper_L(pk,qk),0,1);
      [pw1,qw1] = oper_Lt(xpq);
      pw = pk + Li*pw1;
      qw = qk + Li*qw1;
      [pk,qk] = proj_pair(pw,qw);
      x = proj_bound(xin - lam*oper_L(pk,qk),0,1);
      k = k + 1;
      psnr(k+1) = 20*log10(c/norm(x-xorig,'fro'));
end
disp('PSNR before and after denoising:')
[psnr(1) psnr(K+1) psnr(K+1)-psnr(1)]
map = gray(256);
subplot(221)
imshow(255*xorig,map)
title('original image')
subplot(222)
imshow(255*xin,map)
title('noise contaminated image')
subplot(223)
imshow(255*x,map)
title('denoised image')
subplot(224)
plot(1:1:K+1,psnr)
grid
axis([0 K+1 psnr(1)-3 psnr(K+1)+3])
axis square
title('PSNR profile')
==============
function x = oper_L(p,q)
[t,n] = size(p);
m = t + 1;
zn = zeros(1,n);
zm = zeros(m,1);
pe = [zn;p;zn];
qe = [zm q zm];
x = pe(2:m+1,:) + qe(:,2:n+1) - pe(1:m,:) - qe(:,1:n);
```

```
function [p,q] = oper_Lt(x)
```

```
[m,n] = size(x);
p = x(1:m-1,:) - x(2:m,:);
q = x(:,1:n-1) - x(:,2:n);
```

```
function x = proj_bound(xi,L,U)
xw = max(xi,L);
x = min(xw,U);
```

```
function [r,s] = proj_pair(p,q)
[t,n] = size(p);
m = t + 1;
p1 = p(:,1:n-1);
q1 = q(1:m-1,:);
pq1 = p1.^2 + q1.^2;
mpq = max(1,pq1);
r1 = p1./mpq;
r2 = p(:,n)./max(1,abs(p(:,n)));
r = [r1 r2];
s1 = q1./mpq;
s2 = q(m,:)./max(1,abs(q(m,:)));
s = [s1;s2];
============
```

Next, the FGP algorithm was applied to the image with parameter set-up except that the number of iterations was set to $K = 35$. The algorithm was able to yield a de-noised image with a slightly better PSNR of 27.5136 dB, see Fig. 3.4.

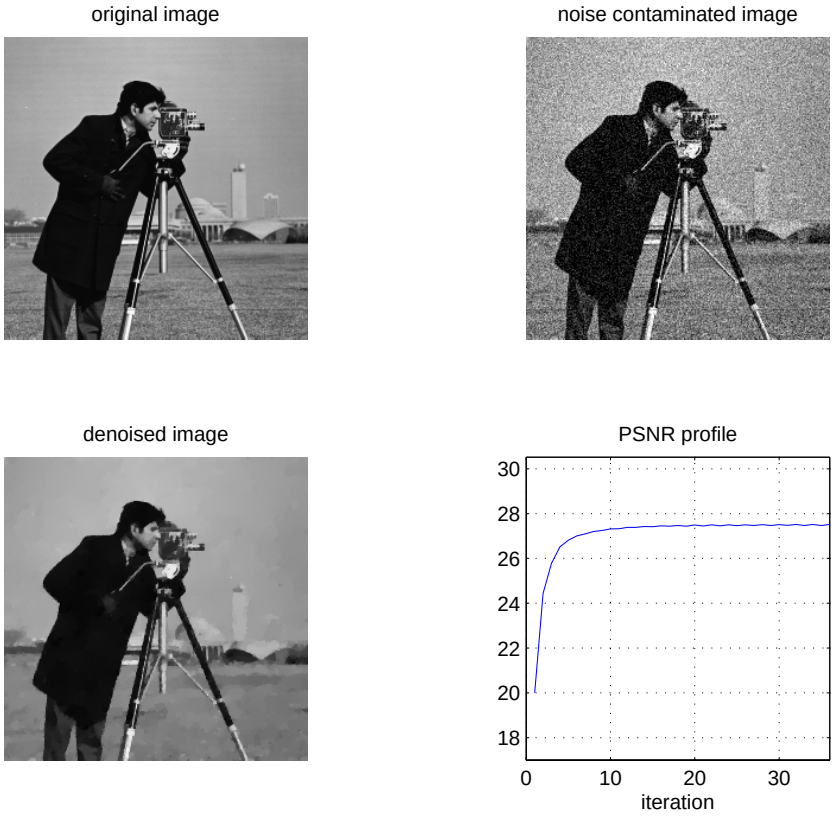


Figure 3.4

```
% Program: fgp_denoise.m
% To remove noise from a corrupted still image by using the
% fast gradient projection (FGP) algorithm described in Sec. 3.2
% of the notes, see pp. 44-45.
% Reference: A. Beck and M. Teboulle, "Fast gradient-based
% algorithms for constrained total variation image denoising
% and deblurring problems," IEEE Trans. Image Processing,
% vol. 18, no. 11, pp. 2419-2434, Nov. 2009.
% Input:
% xorig: original image.
% st: initial random state for noise.
% sig: standard deviation of the noise.
% lam: parameter \lambda.
% K: number of iterations to be performed.
% Output:
% x: de-noised image.
% psnr: profile of peak signal-to-noise ratio in dB.
% Written by W.-S. Lu, University of Victoria.
% Example: [x,psnr] = fgp_denoise(camera256,17,0.1,0.1,35);
function [x,psnr] = fgp_denoise(xorig,st,sig,lam,K)
xorig = xorig/255;
[m,n] = size(xorig);
pk = zeros(m-1,n);
rk = pk;
qk = zeros(m,n-1);
sk = qk;
randn('state',st)
u = sig*randn(m,n);
xin = xorig + u;
k = 0;
tk = 1;
Li = 1/(8*lam);
c = sqrt(m*n);
psnr_before = 20*log10(c/norm(u,'fro'));
psnr = zeros(K+1,1);
psnr(1) = psnr_before;
while k < K,
    xpq = proj_bound(xin - lam*oper_L(rk,sk),0,1);
    [pw1,qw1] = oper_Lt(xpq);
    pw = rk + Li*pw1;
    qw = sk + Li*qw1;
    [pk1,qk1] = proj_pair(pw,qw);
    tk1 = 0.5*(1+sqrt(1+4*tk^2));
    ak = (tk - 1)/tk1;
    rk = pk1 + ak*(pk1-pk);
    sk = qk1 + ak*(qk1-qk);
    tk = tk1;
```

```
    pk = pk1;
    qk = qk1;
    x = proj_bound(xin - lam*oper_L(pk,qk),0,1);
    k = k + 1;
    psnr(k+1) = 20*log10(c/norm(x-xorig,'fro'));
end
disp('PSNR before and after denoising:')
[psnr(1) psnr(K+1) psnr(K+1)-psnr(1)]
map = gray(256);
subplot(221)
imshow(255*xorig,map)
% axis square
title('original image')
subplot(222)
imshow(255*xin,map)
% axis square
title('noise contaminated image')
subplot(223)
imshow(255*x,map)
% axis square
title('denoised image')
subplot(224)
plot(1:1:K+1,psnr)
grid
axis([0 K+1 psnr(1)-3 psnr(K+1)+3])
axis square
title('PSNR profile')
xlabel('iteration')
```

3.3 A Gradient Projection Algorithm for Constrained TV-Based Image De-Blurring

This section introduces a gradient projection algorithm proposed in [17] for image de-blurring. As will become transparent shortly, the algorithm is essentially an *iterative* version of the gradient projection (GP) algorithm described in Sec. 3.2. The basic idea behind the algorithm is to achieve image de-blurring through de-noising of a sequence of related "images". We begin by recalling the de-blurring model given by (3.11). By imposing a constraint that image $\boldsymbol{x}$ belong to a convex set C, the model becomes

$$\underset{\boldsymbol{x}\in C}{\text{minimize}} \quad F(\boldsymbol{x}) = \left\|\mathcal{A}\boldsymbol{x} - \boldsymbol{b}\right\|_F^2 + \lambda \left\|\boldsymbol{x}\right\|_{TV} \tag{3.47}$$

where $\boldsymbol{x}$ is a matrix variable and $\mathcal{A}$ represents a linear blurring operator. As pointed out earlier, the first term in (3.47) is a smooth and convex function of $\boldsymbol{x}$, and the second term is nonsmooth but convex, thus (3.47) fits nicely into the model in (3.1).

Let $\boldsymbol{x}_{k-1}$ be the image obtained in the $(k-1)$th iteration. In the kth iteration, it follows from (3.19) that iterate $\boldsymbol{x}_k$ can be obtained as the minimizer of the convex quadratic function

$$Q_L(\boldsymbol{x}, \boldsymbol{x}_{k-1}) = f(\boldsymbol{x}_{k-1}) + \langle (\boldsymbol{x} - \boldsymbol{x}_{k-1}), \nabla f(\boldsymbol{x}_{k-1}) \rangle + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_{k-1}\|_F^2 + g(\boldsymbol{x}) \tag{3.48}$$

subject to $\boldsymbol{x} \in C$, which implies that $\boldsymbol{x}_k = P_L(\boldsymbol{x}_{k-1})$ where

$$p_L(\boldsymbol{x}_{k-1}) = \arg\min_{\boldsymbol{x} \in C} \left\{ \frac{L}{2} \left\| \boldsymbol{x} - \left(\boldsymbol{x}_{k-1} - \frac{1}{L} \nabla f(\boldsymbol{x}_{k-1}) \right) \right\|_F^2 + \lambda \|\boldsymbol{x}\|_{TV} \right\} \tag{3.49}$$

If we denote

$$\boldsymbol{b}_{k-1} = \boldsymbol{x}_{k-1} - \frac{1}{L} \nabla f(\boldsymbol{x}_{k-1}) \tag{3.50}$$

then (3.49) becomes

$$p_L(\boldsymbol{x}_{k-1}) = \arg\min_{\boldsymbol{x} \in C} \left\{ \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{b}_{k-1}\|_F^2 + \lambda \|\boldsymbol{x}\|_{TV} \right\} \tag{3.51}$$

On comparing (3.51) with (3.25), we see that if the convex set C is $B_{l,u}$ defined in (3.25), then iterate $\boldsymbol{x}_k$ can be found by solving de-noising problem (3.25) with parameter λ replaced by $2\lambda / L$. In other words, de-blurring an image can be treated as a process of iteratively de-noising a sequence of images bk that are varying as the iteration proceeds and are related to the observed (blurred) by (3.50) or more explicitly by (3.53) (see below). Based on this, a gradient projection algorithm for de-blurring (GPB) can be constructed based or algorithm GP or FGP described in Sec. 3.2.

There are two implementation issues that need to be addressed. The first issue is to evaluate $\nabla f(\boldsymbol{x})$.

The function $f(\boldsymbol{x})$ in (3.47) is given by $f(\boldsymbol{x}) = \|\boldsymbol{\mathcal{A}}\boldsymbol{x} - \boldsymbol{b}\|_F^2$ where $\boldsymbol{\mathcal{A}}$ is a *linear* operator, hence the gradient of $f(\boldsymbol{x})$ is given by

$$\nabla f(\boldsymbol{x}) = 2\boldsymbol{\mathcal{A}}^T (\boldsymbol{\mathcal{A}}\boldsymbol{x} - \boldsymbol{b}) \tag{3.52}$$

where $\boldsymbol{\mathcal{A}}^T$ denote the adjoint of operator $\boldsymbol{\mathcal{A}}$, hence $\boldsymbol{b}_{k-1}$ in (3.50) becomes

$$\boldsymbol{b}_{k-1} = \boldsymbol{x}_{k-1} - \frac{2}{L} \boldsymbol{\mathcal{A}}^T (\boldsymbol{\mathcal{A}}\boldsymbol{x}_{k-1} - \boldsymbol{b}) \tag{3.53}$$

The second issue is to estimate the Lipschitz constant L (see (3.2) for its definition). With (3.52), L must satisfy

$$\|2\boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{A}}(\boldsymbol{x} - \boldsymbol{y})\|_F \le L \|\boldsymbol{x} - \boldsymbol{y}\|_F$$

for any $\boldsymbol{x}$ and $\boldsymbol{y}$, hence

$$L = 2 \cdot \max_{\|\boldsymbol{x}\|_F = 1} \|\boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{A}}\boldsymbol{x}\|_F \tag{3.54}$$

The gradient projection algorithm for image de-blurring is outlined below.

Algorithm GPB($\boldsymbol{\mathcal{A}}$, $\boldsymbol{b}$, λ, L, K_n, K_b)

Inputs: blurring operator $\boldsymbol{\mathcal{A}}$, Lipschitz constant L, observed image $\boldsymbol{b}$, regularization parameter λ, iteration number K_n for de-noising algorithm GP, and iteration number K_b for de-blurring.

Output: An optimal solution $\boldsymbol{x}^*$ of (3.47) up to a tolerance.

Step 0 Take $\boldsymbol{x}_0 = \boldsymbol{0}$.

Step k ($k = 1, 2, \ldots, K_b$)

Compute $\boldsymbol{b}_{k-1}$ using (3.53);

Call de-noising algorithm GP($\boldsymbol{b}_{k-1}, 2\lambda / L, K_n$), denote its solution as $\boldsymbol{x}_k$;

Set $\boldsymbol{x}^* = \boldsymbol{x}_{K_b}$.

Example 3.3 The GPB algorithm was applied to image lena of size 256 by 256 which was corrupted by a Gaussian blur with standard deviation 4 followed by an additive white noise with standard deviation 0.001. With $\lambda = 0.0001$, $L = 2$, $K_n = 1$, and $K_b = 2500$, the GPB algorithm produced an image whose PSNR was improved from 23.2151 dB to 29.3245 dB, see Fig. 3.5.

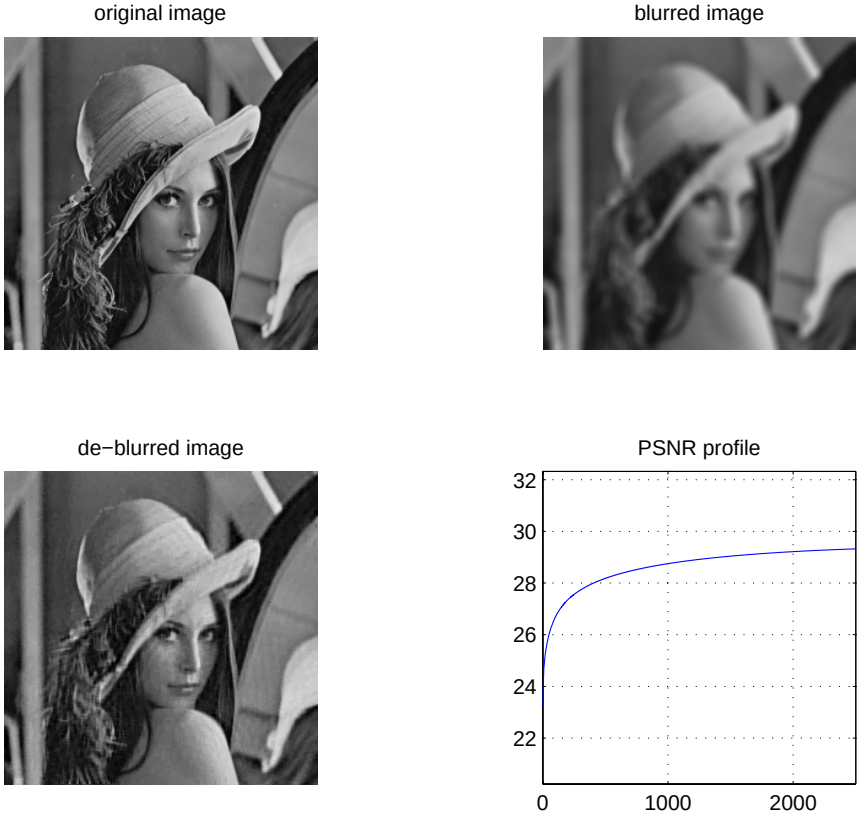


Figure 3.5

```
% Program: gp_deblurr.m
% To remove blurring effect from a corrupted still image by using
% the gradient projection (GPB) algorithm described in Sec. 3.3 of
% the notes, see pp. 49-52.
% Reference: A. Beck and M. Teboulle, "Fast gradient-based
% algorithms for constrained total variation image denoising
% and deblurring problems," IEEE Trans. Image Processing,
% vol. 18, no. 11, pp. 2419-2434, Nov. 2009.
% Input:
% xorig: original image.
% st: initial random state for noise.
% sig_n: standard deviation of the noise.
% sig_b: standard deviation of Gussian blur.
% lam: parameter \lambda.
% L: Lipschitz constant associated with the blurring operator.
```

```
%       For a Gaussian blur, L = 2.
% Kn: number of iterations to be performed for each de-noising subproblem.
% Kb: number of iterations to be performed for de-bluring, which is the
%         number of runs of the de-noising subproblem.
% Output:
% xs: de-blurred image.
% psnr: profile of peak signal-to-noise ratio in dB.
% Written by W.-S. Lu, University of Victoria.
% Example: [xs,psnr] = gp_deblurr(lena256,17,0.001,4,0.0001,2,1,2500);
function [xs,psnr] = gp_deblurr(xorig,st,sig_n,sig_b,lam,L,Kn,Kb)
xorig = xorig/255;
[m,n] = size(xorig);
Hb = fspecial('gaussian',9,sig_b);
Hbt = flipud(fliplr(Hb));
xb = imfilter(xorig,Hb,'replicate');
randn('state',st)
u = sig_n*randn(m,n);
b = xb + u;
Li2 = 2/L;
lw = Li2*lam;
xin = Li2*b;
c = sqrt(m*n);
psnr_before = 20*log10(c/norm(b-xorig,'fro'));
kb = 0;
psnr = zeros(Kb+1,1);
while kb < Kb,
   psnr(1) = psnr_before;
   x = gp_denoise_w(xin,lw,Kn);
   kb = kb + 1;
   yw1 = imfilter(x,Hb,'replicate');
   yw2 = imfilter(yw1-b,Hbt,'replicate');
   xin = x -    Li2*yw2;
   psnr(kb+1) = 20*log10(c/norm(x-xorig,'fro'));
end
xs = x;
disp('PSNR before and after denoising:')
[psnr(1) psnr(Kb+1) psnr(Kb+1)-psnr(1)]
map = gray(256);
subplot(221)
imshow(255*xorig,map)
title('original image')
subplot(222)
imshow(255*b,map)
title('blurred image')
subplot(223)
imshow(255*xs,map)
title('de-blurred image')
```

```
subplot(224)
plot(1:1:Kb+1,psnr)
grid
axis([0 Kb+1 psnr(1)-3 psnr(Kb+1)+3])
axis square
title('PSNR profile')
```

```
% Program: gp_denoise_w.m
% This matlab function is required by function gp_blurr.m
% to solve de-noising subproblems involved.
% Written by W.-S. Lu, University of Victoria.
function x = gp_denoise_w(xin,lam,Kn)
[m,n] = size(xin);
pk = zeros(m-1,n);
qk = zeros(m,n-1);
k = 0;
Li = 1/(8*lam);
while k < Kn,
      xpq = proj_bound(xin - lam*oper_L(pk,qk),0,1);
      [pw1,qw1] = oper_Lt(xpq);
      pw = pk + Li*pw1;
      qw = qk + Li*qw1;
      [pk,qk] = proj_pair(pw,qw);
      x = proj_bound(xin - lam*oper_L(pk,qk),0,1);
      k = k + 1;
end
```

References

[1] S. Mallat, *A Wavelet Tour of Signal Processing*, 3rd ed., San Diego, CA.: Academic Press, 2008.

[2] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal recognition from highly incomplete frequency information," *IEEE Trans. Info. Theory*, vol. 52, no. 2, pp. 489-509, February 2006.

[3] D. Donoho, "Compressed sensing," *IEEE Trans. Info. Theory*, vol. 52, no. 4, pp. 1289-1306, April 2006.

[4] E. J. Candès and T. Tao, "Near optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Trans. Info. Theory*, vol. 52, no. 12, pp. 5406-5425, Dec. 2006.

[5] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, pp. 21-30, March 2008.

[6] E. J. Candès and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969-985, 2007.

[7] E. Candès and J. Romberg, "l_1-MAGIC: Recovery of sparse signals via convex programming," Caltech, October 2005.

[8] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Info. Theory*, vol. 51, no. 12, pp. 4203-4215, Dec. 2005.

[9] E. J. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Comm. Pure Appl. Math*. vol. 59, no. 8, pp. 1207-1223, Aug. 2006.

[10] Y. Nesterov, "Gradient methods for minimizing composite objective function," CORE discussion paper, no. 76, University of Catholic Louvain, Belgium, Sept. 2007.

[11] Y. Nesterov, *Introductory Lectures on Convex Optimization – A Basic Course*, Kluwer Academic Publishers, 2004.

[12] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259-268, 1992.

[13] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math*., vol. 57, pp. 1413-1457, 2004.

[14] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Simul*., vol. 4, pp. 1168-1200, 2005.

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183-202, 2009.

[16] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543-547, 1983. (in Russian)

[17] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation denoising and deblurring problems," *IEEE Trans. Image Processing*, vol. 18, no. 11, pp. 2419-2434, Nov. 2009.

[18] A. Chambolle, "An algorithm for total variation minimization and applications," *J. Math. Imag. Vis*., vol. 20, no. 1, pp. 89-97, 2004.

[19] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[20] J. J. Moreau, "Proximite at dualite dans un espace Hilbertien," *Bull. Soc. Math. France*, vol. 93, pp. 273-299, 1965.

[21] Compressive Sensing Resources, http://www.dsp.ece.rice.edu/cs