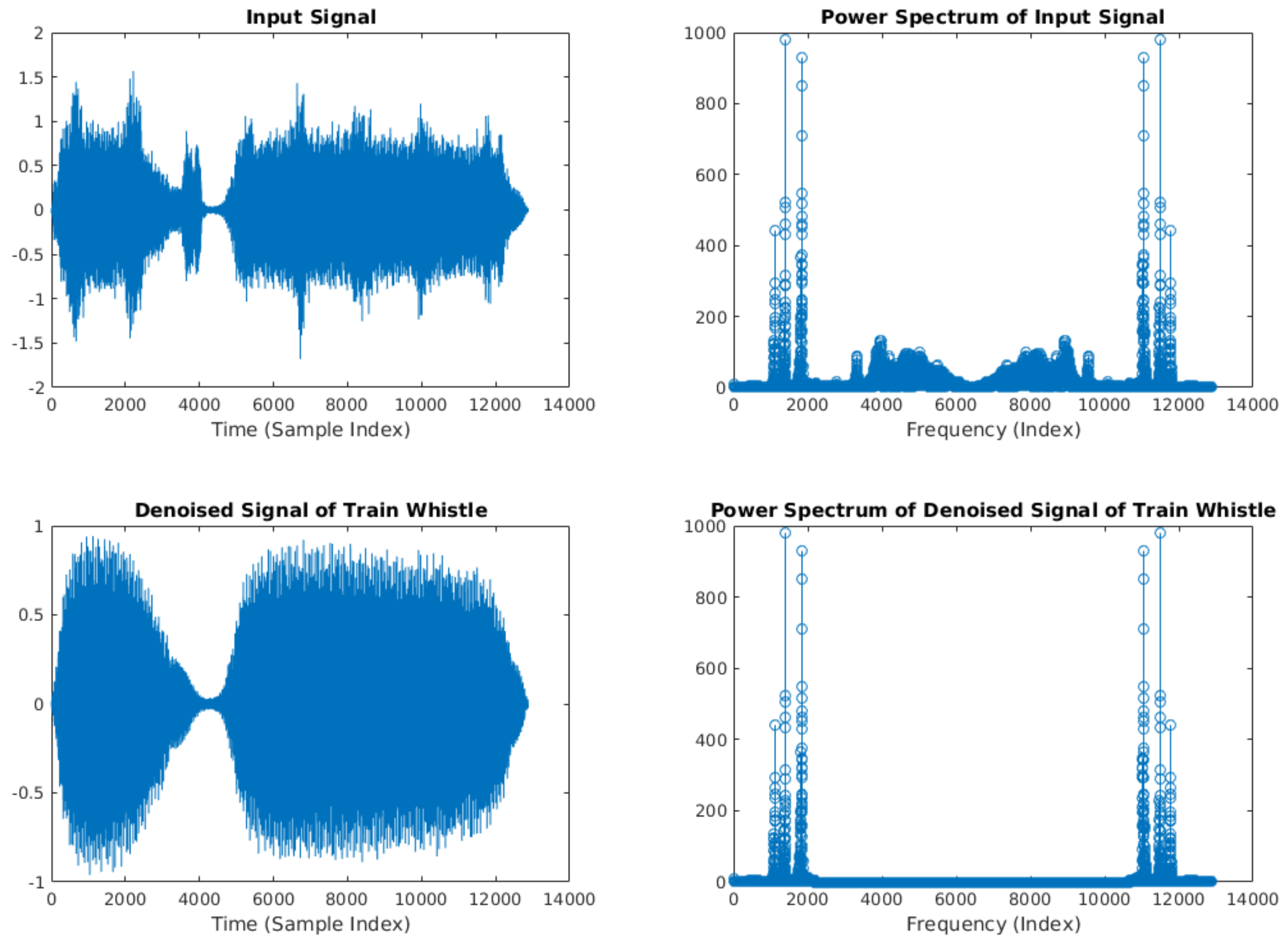
Question 4.

```
% Load Input Data
load('train_bird.mat');
y_len = length(y);
% Plot Input Signal
subplot(2, 2, 1);
plot(y);
title('Input Signal', 'FontSize', 8);
xlabel('Time (Sample Index)');
% Calculate DFT of Input Signal
fre_y = fft(y);
% Plot Power Spectrum of Input Signal
subplot(2, 2, 2);
stem(abs(fre_y));
title('Power Spectrum of Input Signal', 'FontSize', 8);
xlabel('Frequency (Index)');
% Set the limit frequency for picking signal from frequency domain
t_freq = 2200;
% Pick Bird Chirp Signal
fre_bird = fre_y;
fre_bird(1:t_freq) = 0;
fre_bird(y_len-t_freq: y_len) = 0;
y_bird = ifft(fre_bird);
%soundsc(real(y_bird), Fs);
% Pick Train Whistle Signal & Plot
fre_train = fre_y;
fre_train(t_freq+1: y_len-t_freq-1) = 0;
subplot(2, 2, 4);
stem(abs(fre_train));
title('Power Spectrum of Denoised Signal of Train Whistle', 'FontSize', 8);
xlabel('Frequency (Index)');
y_train = ifft(fre_train);
subplot(2, 2, 3);
plot(real(y_train));
title('Denoised Signal of Train Whistle', 'FontSize', 8);
xlabel('Time (Sample Index)');
soundsc(real(y_train), Fs);
```

q4.m


Input Signal
Time (Sample Index)
Power Spectrum of Input Signal
Frequency (Index)
Denoised Signal of Train Whistle
Time (Sample Index)
Power Spectrum of Denoised Signal of Train Whistle
Frequency (Index)

Description:

(i). **Implementation:**

In my code, it firstly load the noised sound data as input signal to array "y" and load the sample rate to an integer "Fs". Then, it plots the input signal and calculate the DFT of input signal by using matlab function "fft". Next, it plots the power spectrum of input signal by calculating the modulus of the DFT of input signal.

In the following, the frequency spectrum of bird chirp signal is picked up by setting all DFT coefficients with index of [1:t_freq] and [y_len – t_freq, y_len] to 0, where "y_len" is the length of input signal array "y", and "t_freq" is the frequency threshold. So, only the DFT coefficients in the middle interval is kept, others are dropped.

Similarly, the frequency spectrum of train whistle signal is picked up by setting all DFT coefficients with index of [t_freq+1, y_len – t_freq-1] to 0. So, only the DFT coefficients on the left-side and right-side are kept, others are dropped.

Finally, rebuild the denoised train whistle by calculate the inverse DFT of picked frequency spectrum of train whistle signal, and plot the denoised signal and its power spectrum.

Results:

The denoised train whistle is almost pure train whistle with only very weak bird chirp;

Similarly, the denoised bird chirp is almost pure bird chirp with only very weak train whistle;

(ii).

Frequency threshold is 2200

(iii).

Similarities:

The overall shapes of both noisy and denoised signal are similar: they both have high magnitude with sample index [1, 3000] and [5000, 13000], and low magnitude with other sample index.

Differences:

The noisy signal looks sharper since it has lots of "peaks", whereas the denoised signal looks smoother since the signal of bird chirp with higher frequency is removed.

Question 5.

(a).

```
function [Y, drop] = Compress(X, tol)
% Block size
BSIZE = 8;
% Get size of input matrix X
[row_cnt, col_cnt] = size(X);
% Pre-allocate memory
Y = zeros(row_cnt, col_cnt);
% Count of dropped blocks
drop_cnt = 0;
% Compress Process
for i = 1:BSIZE:row_cnt
    for j = 1:BSIZE:col_cnt
        % Get a block from input matrix
        in_block = X(i:i+BSIZE-1, j:j+BSIZE-1);
        % Calculate 2D FFT
        out_block = fft2(in_block);
        % Calculate threadhold
        abs_out_block = abs(out_block);
        F_max = max(abs_out_block(:));
        threshold = F_max*tol;
        % Dropping
        for k = 1:BSIZE
            for l = 1:BSIZE
                if abs(out_block(k,l)) < threshold
                    % Satisfy the condition, drop the DFT value of this pixel
                    out_block(k,l) = 0;
                    drop_cnt = drop_cnt + 1;
                end
            end
        end
        % Calculate 2D IFFT
        comp_block = ifft2(out_block);
        % Put the compressed pixel values to output
        Y(i:i+BSIZE-1, j:j+BSIZE-1) = real(comp_block);
    end
end
% calculate drop ratio
drop = double(drop_cnt)/double(row_cnt*col_cnt);
end
```

Compress.m

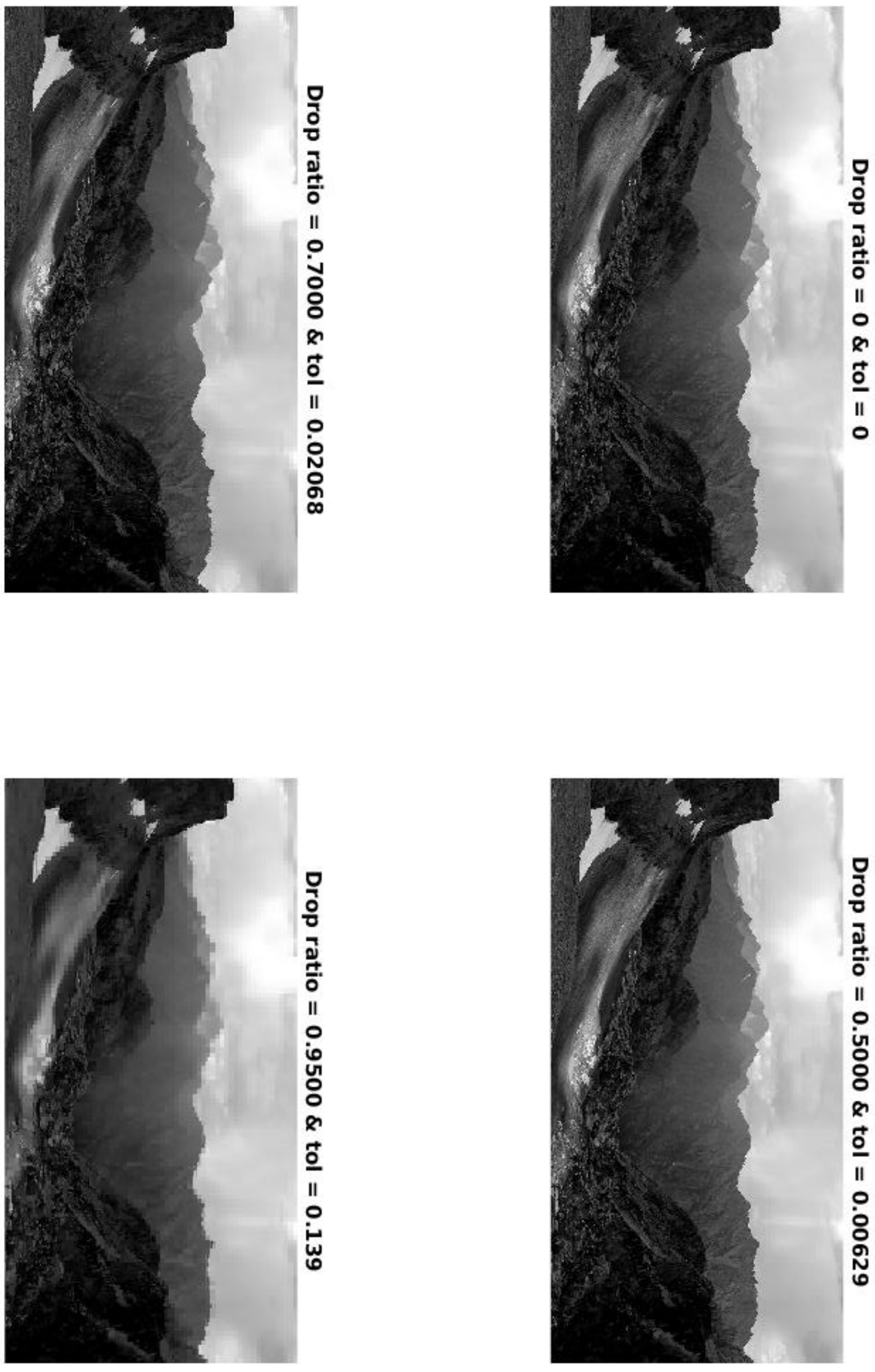
(b).

```
% Pre-allocate
drop_ratio = double(zeros(1, 4));
rel_error = double(zeros(1, 4));

% Read picture & Convert it to double
X = imread('mountainous.jpg');
X = im2double(X);

figure(1);
% Drop = 0
[Y, drop] = Compress(X, 0);
Y(:,:) = min(Y(:,:), 1.0);
Y(:,:) = max(Y(:,:), 0.0);
subplot(2, 2, 1);
imshow(Y);
title('Drop ratio = 0 & tol = 0', 'FontSize', 10);
disp(drop);
% NMSE
drop_ratio(1) = drop;
rel_error(1) = sqrt(mean2((Y-X).^2)/(mean2(X).^2));

% Drop = 0.5
[Y, drop] = Compress(X, 0.00629);
Y(:,:) = min(Y(:,:), 1.0);
Y(:,:) = max(Y(:,:), 0.0);
subplot(2, 2, 2);
imshow(Y);
title('Drop ratio = 0.5000 & tol = 0.00629', 'FontSize', 10);
disp(drop);
% NMSE
drop_ratio(2) = drop;
rel_error(2) = sqrt(mean2((Y-X).^2)/(mean2(X).^2));

% Drop = 0.7
[Y, drop] = Compress(X, 0.02068);
Y(:,:) = min(Y(:,:), 1.0);
Y(:,:) = max(Y(:,:), 0.0);
subplot(2, 2, 3);
imshow(Y);
title('Drop ratio = 0.7000 & tol = 0.02068', 'FontSize', 10);
disp(drop);
% NMSE
drop_ratio(3) = drop;
```

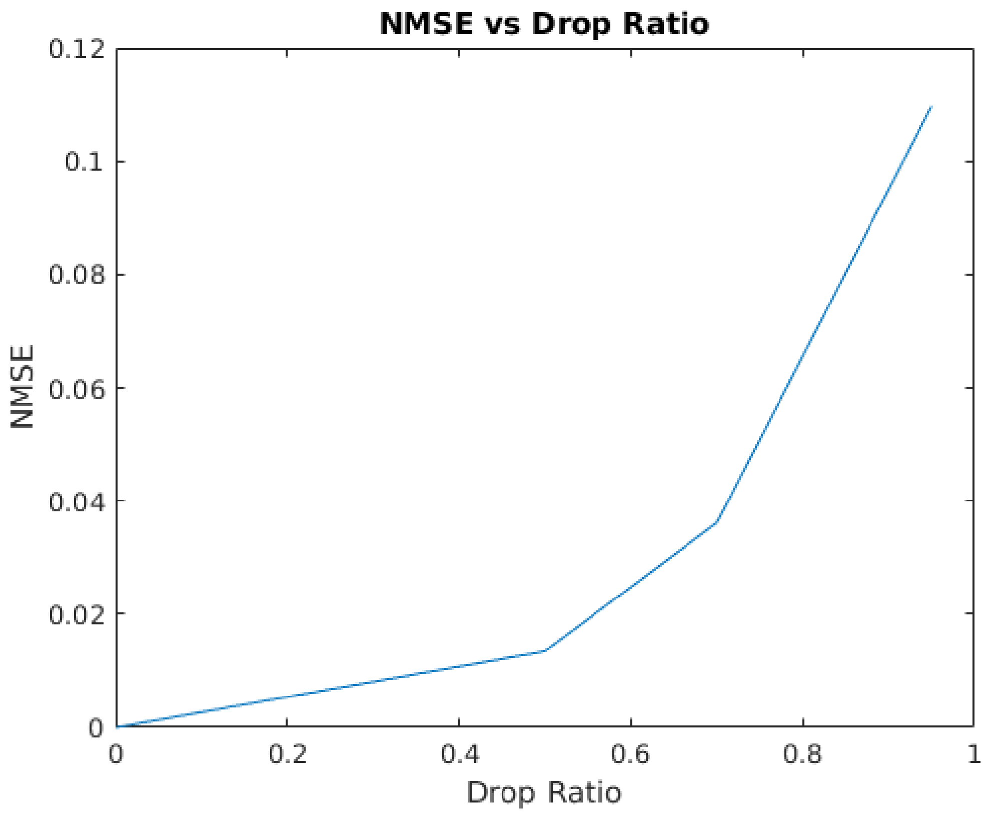
```
rel_error(3) = sqrt(mean2((Y-X).^2)/(mean2(X).^2));

% Drop = 0.95
[Y, drop] = Compress(X, 0.139);
Y(:,:) = min(Y(:,:), 1.0);
Y(:,:) = max(Y(:,:), 0.0);
subplot(2, 2, 4);
imshow(Y);
title('Drop ratio = 0.9500 & tol = 0.139', 'FontSize', 10);
disp(drop);
% NMSE
drop_ratio(4) = drop;
rel_error(4) = sqrt(mean2((Y-X).^2)/(mean2(X).^2));

% Plot NMSE
figure(2);
plot(drop_ratio, rel_error);
title('NMSE vs Drop Ratio');
xlabel('Drop Ratio');
ylabel('NMSE');
```

q5.m

(c).


Drop ratio = 0 & tol = 0
Drop ratio = 0.5000 & tol = 0.00629
Drop ratio = 0.7000 & tol = 0.02068
Drop ratio = 0.9500 & tol = 0.139

(d).



(e).

With increasing drop tolerance, more DFT coefficients are dropped, so the drop ratio is increased, which is the expected result. Also, dropping more coefficients means the difference among pixels in each block is eliminated, and this causes the difference between original image and compressed image increases. So, NMSE increases with increased drop ratio.

Compression is run over each 8*8 block, so the differences among pixels in each block are diminished, but the differences among different blocks become more obvious. Thus, we can see the boundary between each block more obviously, and the quality of compressed images is decreasing with increasing drop tolerance.