Advanced Digital Signal Processing

Computer Experiment #1

Due Date: Nov 14 (Monday), 13:00

Problem 1. Detection of DTMF Signals

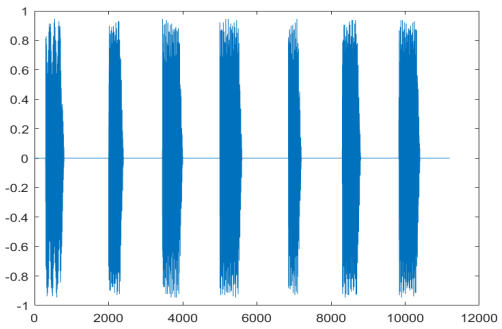


Fig. 1

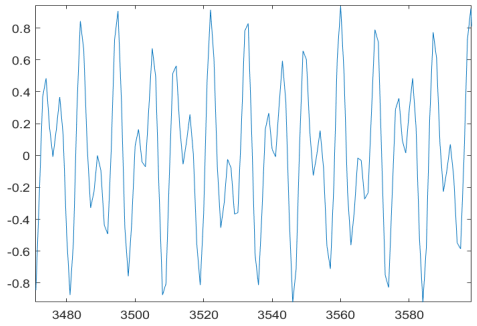


Fig. 2

1. Problem Description: In this problem, you need to detect the DTMF signal as shown in Fig. 1. The DTMF signal contains 7 digits where silence is inserted between consecutive digits. To give you an idea of what this signal looks like, an enlarged waveform plot of some portion of a certain digit segment is shown in Fig. 2. The sampling frequency f_S used is 8000 samples per second, namely, the sampling interval T_S is 1/8 ms. The duration of each digit segment should be at least 40 ms long, which corresponds to 320 samples. The duration of the silence can be different from that of a digit, and should also be at least 40 ms long. Both the durations of signal and silence are random in nature.

2. Goal: Given a recorded sound file (stored in the .wav format), automatically detect the digits that are conveyed by this signal.

3. Method: Apply N-point DFT to detect the seven digits carried by the DTMF signal. The common choice of N for the DTMF application is 205. You should calculate the frequencies of the peak spectral lines carefully. Note that the frequency spacing in the DFT plot is given by f_S / N, i.e., in our setting, the frequency spacing in the DFT plot is 8000/205. By identifying one peak in the low spectral region and the other peak in the high spectral region, you can then determine the corresponding digit that is represented by the signal. Typical DFT spectra of the digits are shown in Figs. 3 and 4. Table 1 lists the nearest k integer values (the index of the DFT spectrum) for the DTMF signal frequencies. Beware that the index of an array starts from 1 in Octave and Matlab, while for the DFT spectrum the index starts from 0.

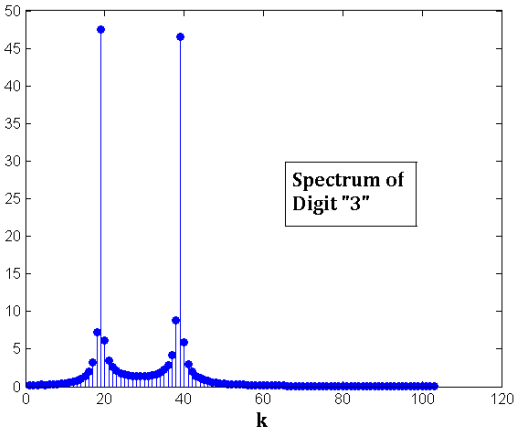


Fig. 3

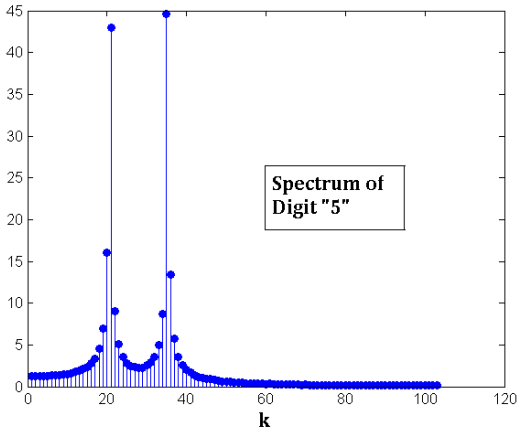


Fig. 4

Table 1 The nearest integer k for the DTMF frequencies

Fundamental Frequency (N=205)	*k value floating-point*	*k value nearest integer*
697.0Hz	17.861	18
770.0Hz	19.731	20
852.0Hz	21.833	22
941.0Hz	24.113	24
1209.0Hz	30.981	31
1336.0Hz	34.235	34
1477.0Hz	37.848	38
1633.0Hz	41.846	42

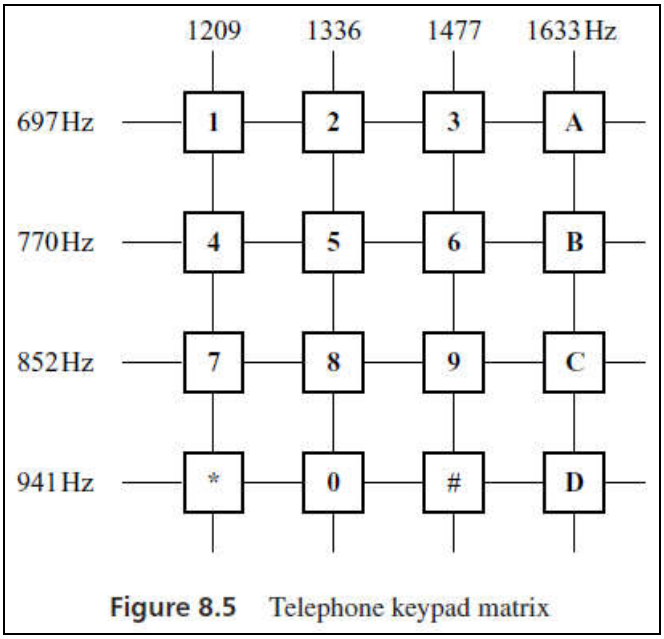


Figure 8.5 Telephone keypad matrix

4. Useful MATLAB Functions:

```
clear all;
[s, fs, nbits] = audioread(InFile); % fs is the sampling rate
sound(s, fs); % play the sound
Fs1 = fft(s1); % take length-N DFT
MFs1 = abs(Fs1); % magnitude spectrum
figure; stem(MFs1(1:round(N/2)),'filled'); % show the spectrum
disp('The numbers are');
disp(output); % display the result
```

5. Key Issues to Be Explored and Discussed:

(1) effect of N

(2) unknown signal durations and silence durations, both random

(3) noise effect in the signal

(4) the **Goertzel algorithm** (a faster method commonly used in practice)

Problem 2. Voice Privacy

1. Problem: Speech privacy techniques are used to scramble clear speech into an *unintelligible* signal in order to avoid *eavesdropping*. One important transform domain scrambler uses the fast Fourier transform (FFT). The block diagram of the FFT scrambler is shown in Fig. 1. At the transmitter, an analog speech signal is filtered, sampled at 8 kHz, digitized by an A/D converter, and then fed to the FFT processor. The FFT coefficients generated by the FFT processor are permuted and fed to the inverse FFT (IFFT) processor. The output of the IFFT processor is D/A converted, multiplexed with the synchronization signal, and then sent out of the transmitter. At the receiver, the received signal is equalized to compensate for the group delay of the transmission channel, and the scrambled speech signal is extracted by a bandpass filter (BPF4). It is then A/D converted and fed to the same type of processor used for the transmitter, except that the FFT coefficients are inversely permuted. The frame and sample timing signals necessary for descrambling are derived from the synchronization signal, which is obtained by filtering the received signal through a bandpass filter (BPF6).

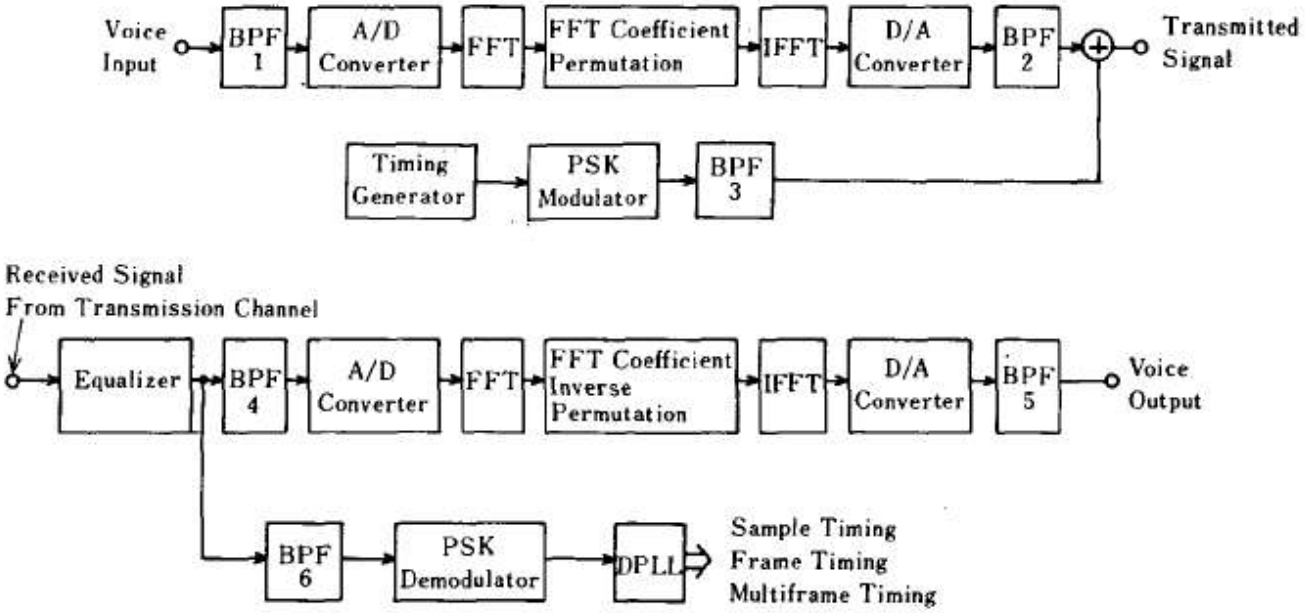
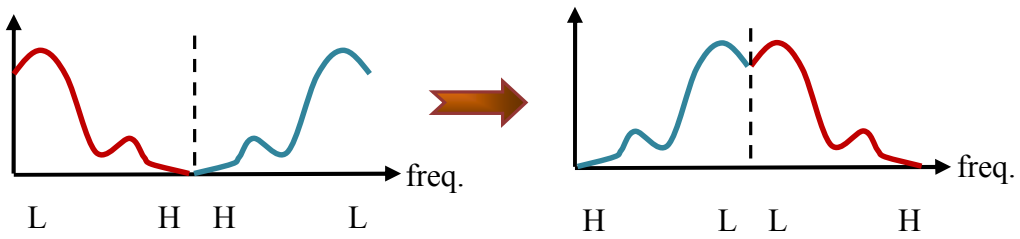


Fig. 1. Block diagram of the FFT scrambler.

Fig. 2. shows an example to illustrate the idea of frequency-domain scrambling. In this problem, we consider a particularly simple FFT coefficient permutation scheme where the low frequency bins and high frequency bins are swapped with each other. To be more specific, suppose the frame length N is 256 (i.e., the number of samples in a FFT block) and X(k) represents the FFT of x(n). Then the permutation of FFT coefficients is given by

$$\begin{cases} X_{\text{new}}(k) = X(k+\frac{N}{2}), & \text{for} \quad 0 \le k \le \frac{N}{2}-1 \\ X_{\text{new}}(k+\frac{N}{2}) = X(k), & \text{for} \quad 0 \le k \le \frac{N}{2}-1 \end{cases} \tag{1}$$



Note that before taking the FFT, the input signal is first divided into frames of N samples each, where the frame length N is typically chosen to be **256** in this application.

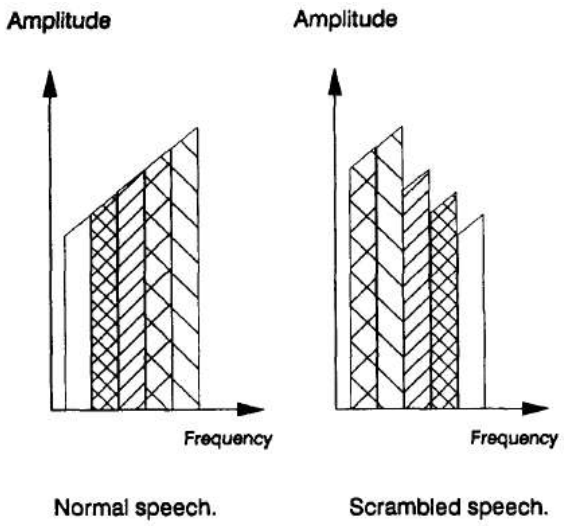


Figure 2: *Frequency domain scrambling*

2. Goal: Given scrambled sound files (stored in the .wav format), recover the original signals.

3. Method 1: Frequency-Domain Scrambling

(1) Segment the input signal (test signal 1) into non-overlapped frames where each frame contains 256 samples.
(2) Take 256-point FFT of each frame.
(3) Apply Eq. (1) to swap low-frequency and high-frequency FFT coefficients.
(4) Take 256-point inverse FFT of each frame.
(5) Pad some number of zeros to the final frame so that its length is also 256.
(6) When all frames are recovered, write the descrambled signal to a file in the .wav format.
Beware that the index of an array starts from 1 in Octave and Matlab, while for the DFT spectrum the index starts from 0. You can listen to the sound files to compare the results before and after the descrambling process.

4. Method 2: Combined Time-Domain and Frequency-Domain Scrambling

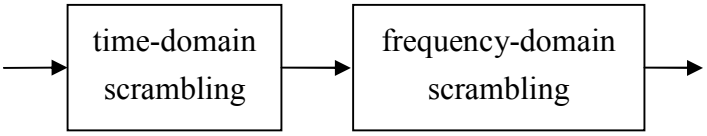


Fig. 3 The scrambling scheme used to generate the second test signal.

For the second test signal, a slightly more complicated scheme for scrambling the speech is adopted. In addition to the frequency-domain scrambling, a time-domain scrambling step is applied beforehand as shown in Fig. 3. As a consequence, to descramble the scrambled speech, the corresponding processing procedures should be carried out in the reverse order, as illustrated in Fig. 4.

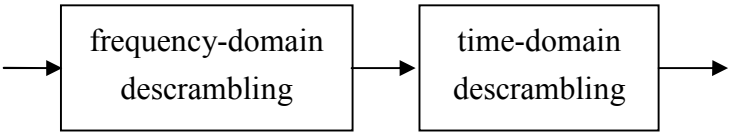


Fig. 4 The two-step descrambling used in the descrambler.

The frequency-domain scrambling/descrambling technique for Figs. 3 and 4 are the same as those used in generating test signal 1. The time-domain descrambling is based on the idea of *permutation* of time-domain samples, and in particular, we consider an easy scheme to perform *time-reversal* of each small group of speech frames. Specifically, the number of speech frames in a multi-frame group is chosen to be 8. The last few frames which are not long enough to form a multi-frame group are left unaltered. The time-domain scrambling is detailed in Fig. 5.

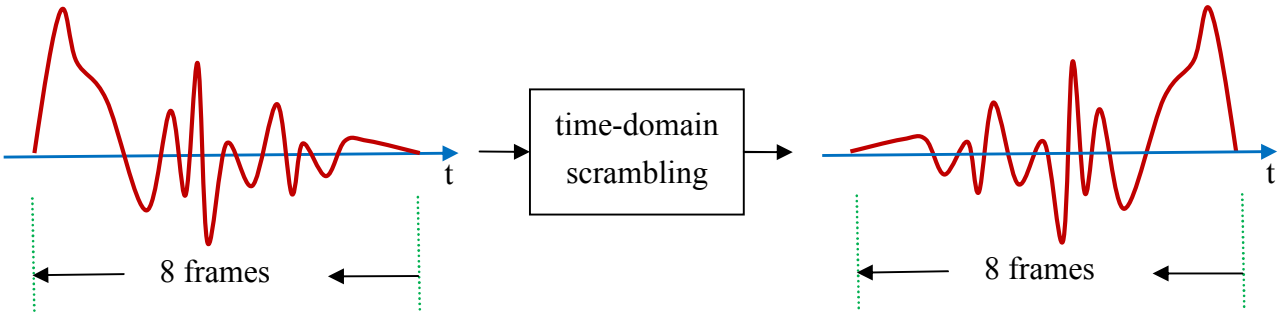


Fig. 5 Graphical illustration of the time-reversal operation used in time-domain scrambling where a group of 8 frames is treated as a processing unit.

5. Useful MATLAB Functions:

```
clear all;
[xin, fs, num_bits] = audioread('test2b_8k_fft_scrambled.wav');
%fs is the sampling rate
Lx = length(x); % total length of the speech signal
FN = ceil(Lx/N); % number of frames
nz = N * FN - Lx; % number of zeros to be appended to the last frame
x = [x; zeros(nz,1)]; % the augmented input with padded zeros
audiowrite(x_descrb, fs, num_bits, 'your file name.wav'); % save the
audio
```

6. Key Issues to Be Explored and Discussed:

(1) computational burden

(2) level of security

(3) system latency

7. Report:

- The entire report should encompass **both Problems 1 and 2**.
- The guideline about writing your project report and submitting your work is given in a separate file, entitled "Notes on Submitting your ADSP Project Report" or "ADSP Project 報告上傳須知."
- The complete report package, including report, programs, and output signals, should be mailed to

  dipwork405@gmail.com

  prior to the deadline.