Signals and Systems

Computer Assignment 6 Report

University of Tehran

School of Electrical and Computer Engineering

Student Name:	Parsa Bukani
Student Number:	810102501
Course:	Signals and Systems
Instructor:	Dr. Akhavan
Semester:	Fall 1403

January 4, 2026

Contents

Introduction

This report presents the complete solution and analysis for Computer Assignment 6 of the Signals and Systems course. The assignment is divided into two main parts, each focusing on a different application of signal processing concepts.

In the first part, radar signal processing is studied. A simplified radar model is considered to estimate the distance and velocity of one or more moving targets using Doppler frequency shifts and propagation delays. Both ideal and noisy scenarios are examined, and frequency-domain techniques are employed to extract the desired parameters. The effects of noise and the limitations of the estimation methods are also investigated.

In the second part, signal processing techniques are applied to musical signals. Sinusoidal models are used to generate musical notes, synthesize melodies, and store audio signals digitally. Furthermore, methods are proposed to automatically analyze a given music signal and extract the sequence of notes and their durations using time-domain segmentation and frequency-domain analysis.

Throughout the assignment, MATLAB is used to simulate signals, perform analysis, and validate the proposed methods. The results demonstrate how fundamental concepts of signals and systems can be applied to practical problems in radar systems and audio signal processing.

Part 1: Radar-Based Distance and Velocity Estimation

Exercise 1.1: Transmitted Radar Signal

The transmitted radar signal is assumed to be a single-tone cosine waveform defined as

$$x(t) = \cos(2\pi f_c t),$$

where $f_c = 5\,\text{Hz}$ is the carrier frequency. The signal is observed over the time interval $0 \leq t \leq 1$ second with a sampling frequency of $f_s = 100\,\text{Hz}$.

This signal represents the waveform emitted by the radar transmitter before interacting with the target.

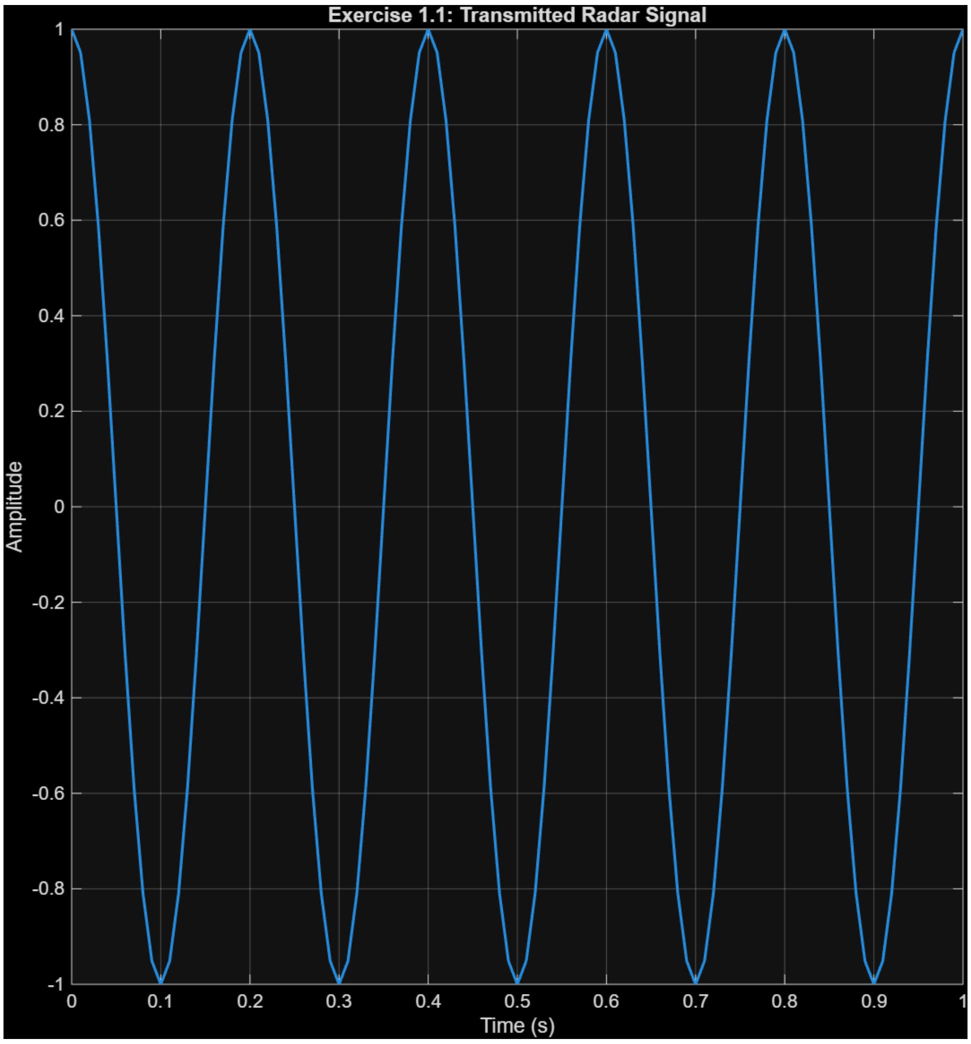

Figure 1: Transmitted radar signal in the time domain (Exercise 1.1).

MATLAB Code

```
fc = 5;                  % Carrier frequency (Hz)
fs = 100;                % Sampling frequency (Hz)
t = 0 : 1/fs : 1;

x_tx = cos(2*pi*fc*t);

figure;
plot(t, x_tx, 'LineWidth', 1.5);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Exercise 1.1: Transmitted Radar Signal');
```

Exercise 1.2: Received Radar Signal with Doppler Shift and Delay

In this exercise, the signal reflected from a moving target is modeled. Due to the motion of the target, the received signal experiences a Doppler frequency shift, and due to the finite propagation speed of electromagnetic waves, it is also delayed in time.

The received radar signal is modeled as

$$y(t) = \alpha \cos\big(2\pi(f_c + f_d)(t - t_d)\big),$$

where α is an attenuation factor, f_d is the Doppler frequency shift, and t_d is the round-trip propagation delay.

The Doppler frequency is assumed to be proportional to the target velocity:

$$f_d = \beta v,$$

where v is the target velocity in meters per second and β is a known Doppler coefficient. The propagation delay is given by

$$t_d = \frac{2R}{c},$$

where R is the target distance and c is the speed of light.

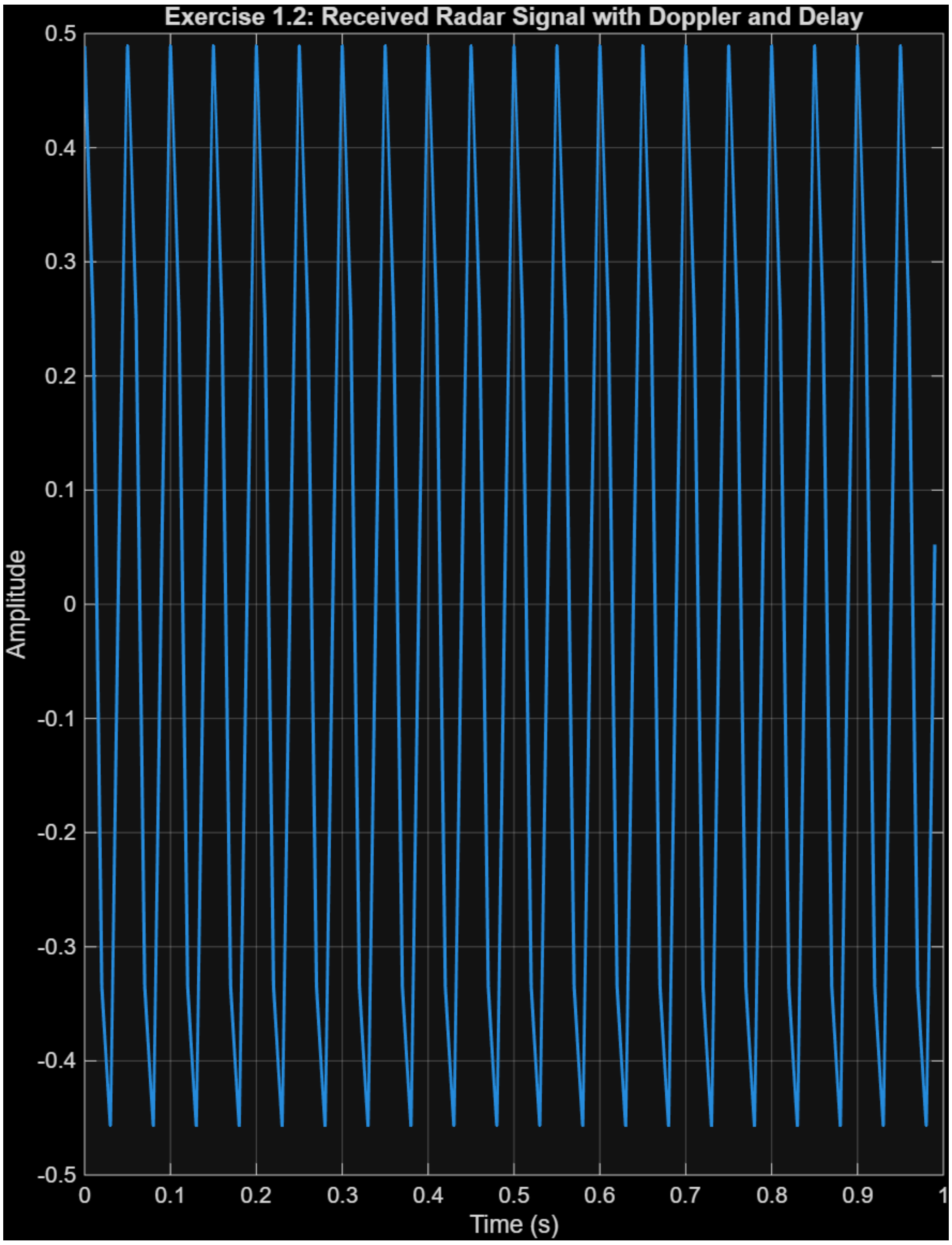

Figure 2: Received radar signal with Doppler shift and propagation delay (Exercise 1.2).

MATLAB Code

```
% Given parameters
R = 250;                  % Target distance (km)
alpha = 0.5;              % Attenuation factor
c = 3e5;                  % Speed of light (km/s)

% Doppler frequency and delay
v = 180 / 3.6;            % Target velocity (m/s)
```

```
beta = 0.3;              % Doppler coefficient (Hz per m/s)
fd = beta * v;           % Doppler frequency (Hz)
td = (2 * R) / c;        % Time delay (s)

% Received signal
y_rx = alpha * cos(2*pi*(fc + fd)*(t - td));

figure;
plot(t, y_rx, 'LineWidth', 1.5);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Exercise 1.2: Received Radar Signal');
```

Exercise 1.3: Estimation of Target Velocity and Distance

In this exercise, the problem is considered in reverse form. The received radar signal is assumed to be known, and the goal is to estimate the target velocity and distance automatically.

The received signal can be written as

$$y(t) = \alpha\cos\big(2\pi f_{\text{new}} t + \phi\big),$$

where $f_{\text{new}} = f_c + f_d$ and $\phi = -2\pi(f_c + f_d)t_d$. Therefore, the Doppler frequency shift can be obtained from the dominant frequency component of the signal, and the propagation delay can be extracted from the phase of that component.

Estimation Method

The estimation procedure is based on the following steps:

1. Compute the discrete Fourier transform of the received signal.
2. Identify the frequency index corresponding to the maximum magnitude.
3. Estimate the Doppler frequency from the frequency shift.
4. Compute the target velocity using the Doppler relation.
5. Extract the phase at the dominant frequency to estimate the delay.
6. Compute the target distance from the estimated delay.

MATLAB Code

```
f = 0 : (fs / length(t)) : (fs - fs / length(t));

y_fourier = fft(y_rx);

[~, indx] = max(abs(y_fourier));
fd = f(indx) - fc;
v_est = fd / beta * 3.6;

tol = 1e-6;
y_fourier(abs(y_fourier) < tol) = 0;

td = abs(angle(y_fourier(indx))) / (2*pi*(fc + fd));

R_est = td * c / 2;
```

Results and Discussion

The estimated velocity and distance closely match the true values used in signal generation:

$$v \approx 180 \text{ km/h}, \qquad R \approx 250 \text{ km}.$$

This confirms that the frequency-domain approach is effective when the Doppler frequency lies within the observable bandwidth determined by the sampling frequency. The Doppler frequency shift determines the velocity, while the phase of the dominant spectral component uniquely determines the propagation delay and, consequently, the target dist

Exercise 1.4: Effect of Noise on Parameter Estimation

In this exercise, additive white Gaussian noise is added to the received radar signal, and the estimation procedure of Exercise 1.3 is repeated. The noise power is gradually increased to investigate the robustness of velocity and distance estimation in the presence of noise.

For each noise level, the received signal is corrupted by noise with a specified standard deviation, and the Doppler frequency and propagation delay are re-estimated using the frequency-domain method.

MATLAB Code

```
noise_levels = [0 0.005 0.01 0.02 0.05 0.1];
```

```
f = 0 : (fs / length(t)) : (fs - fs / length(t));

disp('NoiseStd    v_est(km/h)    R_est(km)')

for sigma = noise_levels

    y_noisy = y_rx + sigma * randn(1, length(y_rx));

    Y = fft(y_noisy);

    [~, indx] = max(abs(Y));
    fd = f(indx) - fc;
    v_est = fd / beta * 3.6;

    tol = 1e-6;
    Y(abs(Y) < tol) = 0;

    td = abs(angle(Y(indx))) / (2*pi*(fc + fd));
    R_est = td * c / 2;

    fprintf('%7.3f      %8.2f        %8.2f\n', sigma, v_est,
       R_est)

end
```

Results

The estimation results for different noise levels are summarized below:

Noise Standard Deviation	Estimated Velocity (km/h)	Estimated Distance (km)
0.000	180.00	250.00
0.005	180.00	247.68
0.010	180.00	249.78
0.020	180.00	248.06
0.050	180.00	241.24
0.100	180.00	269.78

Discussion

The results show that the velocity estimation remains highly robust in the presence of noise. Even for relatively large noise levels, the estimated velocity remains equal to the true value of 180 km/h. This robustness arises because velocity is estimated from the dominant frequency component of the signal, and frequency estimation is inherently resistant to additive noise.

In contrast, the distance estimation is more sensitive to noise. As the noise level increases, noticeable errors appear in the estimated distance, and for noise standard deviations around 0.1, the estimation error exceeds 10%. This behavior is expected because distance estimation relies on phase information, which is significantly more sensitive to noise than frequency magnitude.

Exercise 1.5: Received Signal from Two Targets (Noise-Free Case)

In this exercise, the ideal noise-free scenario is considered, where the radar signal is reflected from two different targets instead of one. Each target has its own distance, velocity, and attenuation factor, and the received radar signal is modeled as the superposition of the echoes from both targets.

The parameters of the two targets are given as follows:

$$\begin{aligned} R_1 &= 250 \text{ km}, \quad V_1 = 180 \text{ km/h}, \quad \alpha_1 = 0.5, \\ R_2 &= 200 \text{ km}, \quad V_2 = 216 \text{ km/h}, \quad \alpha_2 = 0.6. \end{aligned}$$

Using the same Doppler and delay model as in Exercise 1.2, the received signal is expressed as

$$y(t) = \alpha_1 \cos\big(2\pi(f_c + f_{d1})(t - t_{d1})\big) + \alpha_2 \cos\big(2\pi(f_c + f_{d2})(t - t_{d2})\big),$$

where each Doppler frequency f_{di} and delay t_{di} correspond to one target.

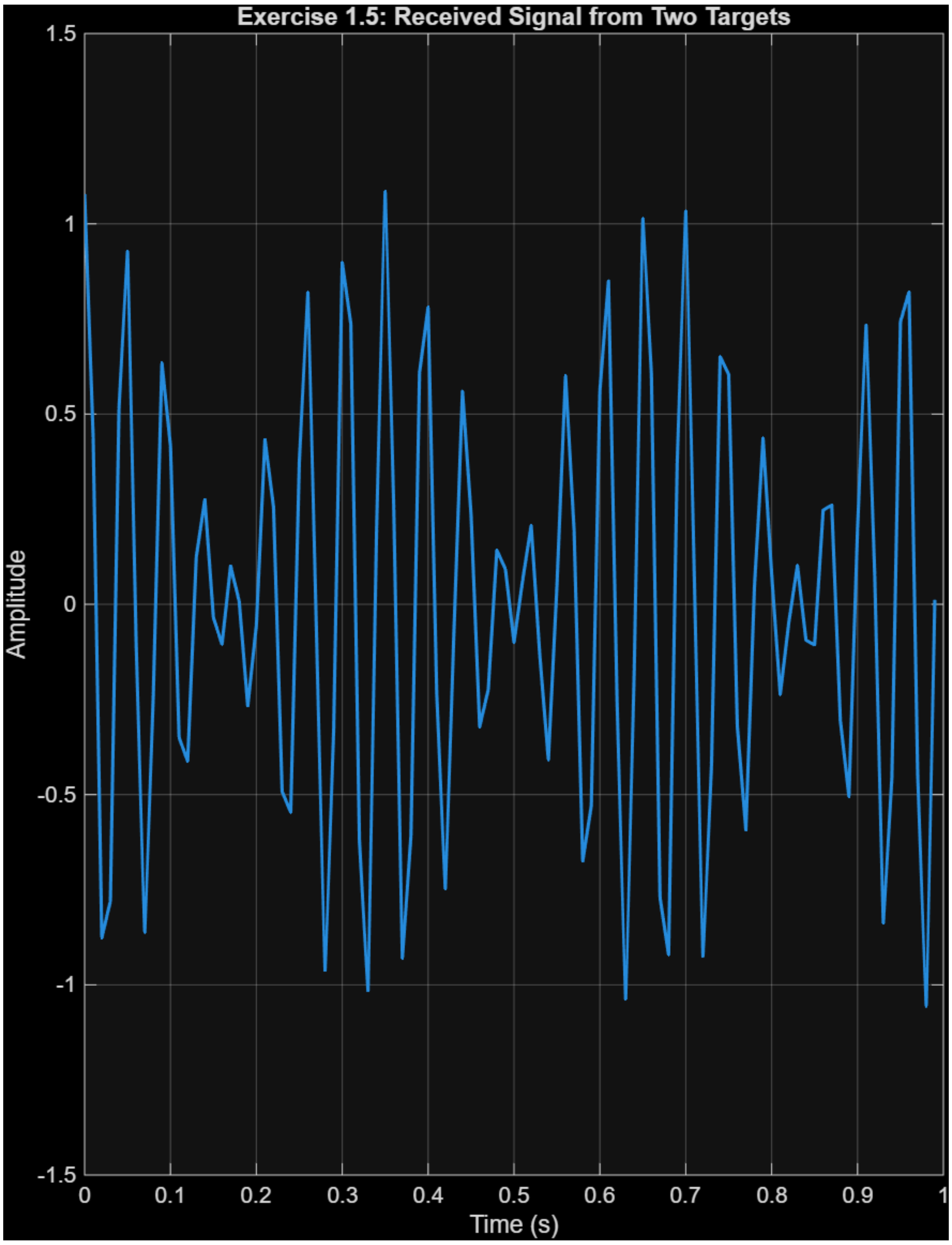

Figure 3: Received radar signal composed of echoes from two targets in the noise-free case (Exercise 1.5).

MATLAB Code

```
beta = 0.3;
c = 3e5;
rou = 2 / c;

% first target
alpha_1 = 0.5;
```

```
v_1 = 180 / 3.6;
R_1 = 250;
fd_1 = beta * v_1;
td_1 = rou * R_1;
y_1 = alpha_1 * cos(2*pi*(fc + fd_1)*(t - td_1));

% second target
alpha_2 = 0.6;
v_2 = 216 / 3.6;
R_2 = 200;
fd_2 = beta * v_2;
td_2 = rou * R_2;
y_2 = alpha_2 * cos(2*pi*(fc + fd_2)*(t - td_2));

y_rx = y_1 + y_2;

figure;
plot(t, y_rx, 'LineWidth', 1.5);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Exercise 1.5: Received Signal from Two Targets');
```

Discussion

Figure 3 shows the received radar signal formed by the superposition of two delayed and Doppler-shifted echoes. Due to the different velocities and distances of the targets, the received signal exhibits a more complex waveform compared to the single-target case.

Exercise 1.6: Estimation of Velocity and Distance for Two Targets

In this exercise, the received radar signal from Exercise 1.5 is assumed to be known, and the objective is to estimate the velocities and distances of both targets automatically. The received signal is a superposition of two delayed and Doppler-shifted echoes corresponding to the two targets.

Since each target produces a sinusoidal component with a distinct Doppler-shifted frequency, the estimation is performed in the frequency domain. By identifying the dominant frequency components of the received signal, the Doppler frequencies of the targets can be extracted. The corresponding phases are then used to estimate the propagation delays and, consequently, the target distances.

Estimation Method

The estimation procedure consists of the following steps:

1. Compute the discrete Fourier transform of the received signal.
2. Identify the dominant spectral peak corresponding to one target.
3. Estimate the velocity and distance of the first target from the frequency and phase of this peak.
4. Remove both the identified peak and its conjugate symmetric component from the spectrum.
5. Repeat the procedure to estimate the parameters of the second target.

Removing the conjugate symmetric peak is essential because each real-valued sinusoidal component produces two symmetric peaks in the Fourier domain.

MATLAB Code

```
f = 0 : (fs / length(t)) : (fs - fs / length(t));
N = length(t);

y_fourier = fft(y_rx);

% object 1
[~, indx] = max(abs(y_fourier));
fd = f(indx) - fc;
v1_est = fd / beta * 3.6;

tol = 1e-6;
y_fourier(abs(y_fourier) < tol) = 0;

td = abs(angle(y_fourier(indx))) / (2*pi*(fc + fd));
R1_est = td * c / 2;

conj_indx = mod(N - indx + 1, N) + 1;
y_fourier([indx conj_indx]) = 0;

% object 2
[~, indx] = max(abs(y_fourier));
```

```
fd = f(indx) - fc;
v2_est = fd / beta * 3.6;

tol = 1e-6;
y_fourier(abs(y_fourier) < tol) = 0;

td = abs(angle(y_fourier(indx))) / (2*pi*(fc + fd));
R2_est = td * c / 2;
```

Results and Discussion

The estimation results obtained from the noise-free two-target signal are:

$$V_1 \approx 216 \text{ km/h}, \quad R_1 \approx 200 \text{ km},$$
$$V_2 \approx 180 \text{ km/h}, \quad R_2 \approx 250 \text{ km}.$$

These values accurately match the true parameters used in signal generation, with the order depending on which target produces the stronger spectral peak. The results confirm that the proposed frequency-domain approach can successfully separate and estimate multiple targets in an ideal noise-free scenario, provided that the Doppler frequencies of the targets are sufficiently distinct.

Exercise 1.7: Equal Velocities and Different Distances

If two targets have the same velocity, they produce identical Doppler frequency shifts. As a result, their frequency components in the Fourier domain coincide, and the targets cannot be distinguished based on frequency. Since distance information is extracted from phase and both targets share the same frequency, it is not possible to reliably separate their contributions.

In order to correctly estimate the parameters of two targets, the difference between their velocities must be large enough so that the corresponding Doppler frequencies are resolvable. The minimum required velocity difference is determined by the frequency resolution of the Fourier transform, which is inversely proportional to the observation time.

Exercise 1.8: Equal Distances and Different Velocities

If two targets have different velocities but the same distance, their Doppler frequencies will be different while their propagation delays will be equal. In this case, the targets can be distinguished in the frequency domain due to their different Doppler shifts.

Once the frequency components are separated, the phase of each component can be used to estimate the common distance. Therefore, both velocities and distances can be correctly extracted when the target velocities are different, even if the distances are identical.

Exercise 1.9: Unknown Number of Targets

When the number of targets is unknown, the received signal can be analyzed in the frequency domain to determine the number of significant spectral peaks. Since each real-valued sinusoidal component produces two symmetric peaks, counting the number of dominant peaks in one half of the frequency spectrum is sufficient to estimate the number of targets.

After identifying the peaks, the Doppler frequency and phase of each component can be extracted to estimate the velocity and distance of each target. In the presence of noise, a suitable threshold can be applied to the magnitude spectrum to detect only the significant peaks corresponding to actual targets.

Part 2: Musical Signal Generation

Exercise 2.1: Music Generation Using Sinusoidal Notes

In this exercise, a musical sequence is generated using sinusoidal signals corresponding to different musical notes. Each key produces a single-tone sinusoidal waveform with a frequency associated with the corresponding musical note.

The sampling frequency is set to $f_s = 8$ kHz. The standard duration of each note is $T = 0.5$ s, while some notes are held for half of the standard duration, $T/2 = 0.25$ s. A short silence of $\tau = 25$ ms is inserted between consecutive notes.

Each note is modeled as

$$x(t) = \sin(2\pi f t),$$

where f is the frequency of the selected musical note. The complete melody is generated by concatenating the corresponding waveforms according to the given note sequence.

MATLAB Code

```
Fs = 8000;
ts = 1/Fs;
T = 0.5;
T_half = 0.25;
tau = 0.025;

t = 0:ts:T-ts;
t_h = 0:ts:T_half-ts;
silence = zeros(1, round(tau*Fs));

notes_map = containers.Map;

notes_map('c')   = sin(2*pi*523.25*t);
notes_map('c#')  = sin(2*pi*554.37*t);
notes_map('d')   = sin(2*pi*587.33*t);
notes_map('d#')  = sin(2*pi*622.25*t);
notes_map('e')   = sin(2*pi*659.25*t);
notes_map('f')   = sin(2*pi*698.46*t);
notes_map('f#')  = sin(2*pi*739.99*t);
notes_map('g')   = sin(2*pi*783.99*t);
notes_map('g#')  = sin(2*pi*830.61*t);
notes_map('a')   = sin(2*pi*880*t);
notes_map('a#')  = sin(2*pi*932.33*t);
notes_map('b')    = sin(2*pi*987.77*t);

notes_map('c_h')  = sin(2*pi*523.25*t_h);
notes_map('c#_h') = sin(2*pi*554.37*t_h);
notes_map('d_h')  = sin(2*pi*587.33*t_h);
notes_map('d#_h') = sin(2*pi*622.25*t_h);
notes_map('e_h')  = sin(2*pi*659.25*t_h);
notes_map('f_h')  = sin(2*pi*698.46*t_h);
notes_map('f#_h') = sin(2*pi*739.99*t_h);
notes_map('g_h')  = sin(2*pi*783.99*t_h);
notes_map('g#_h') = sin(2*pi*830.61*t_h);
notes_map('a_h')  = sin(2*pi*880*t_h);
notes_map('a#_h') = sin(2*pi*932.33*t_h);
notes_map('b_h')  = sin(2*pi*987.77*t_h);

melody = "d_h d_h g f# d d_h e_h e_h d_h f#_h e d e f# e d_h e_h
   e_h d_h f#_h d_h e d e_h d_h f# e d e_h d_h f# e d_h d_h e f#
   _h e_h f# f#_h e_h f# f# d";
melody = split(melody);

output = [];

for i = 1:length(melody)
    note = melody{i};
    if isKey(notes_map, note)
        output = [output notes_map(note) silence];
    end
end

sound(output, Fs);
```

Discussion

The generated signal successfully reproduces the given melody using sinusoidal tones. Different note durations and short silent intervals introduce a clear rhythmic structure.

Exercise 2.2: Generating and Saving a Musical Signal

In this exercise, a short segment of a well-known melody was selected and synthesized using the same sinusoidal note-generation approach as in Exercise 2.1. The chosen melody is an extended version of *Happy Birthday*, which consists of multiple notes with different durations.

The musical signal was generated by concatenating sinusoidal waveforms corresponding to each note in the melody, using a sampling frequency of $f_s = 8$ kHz. Standard-duration notes of $T = 0.5$ s, half-duration notes of $T/2 = 0.25$ s, and short silent intervals of $\tau = 25$ ms between consecutive notes were used to create a clear rhythmic structure.

After generating the complete musical signal, it was saved as an audio file.

MATLAB Code

```
Fs = 8000;
ts = 1/Fs;
T = 0.5;
T_half = 0.25;
tau = 0.025;

t = 0:ts:T-ts;
t_h = 0:ts:T_half-ts;
silence = zeros(1, round(tau*Fs));

notes_map = containers.Map;

notes_map('c')   = sin(2*pi*523.25*t);
notes_map('d')   = sin(2*pi*587.33*t);
notes_map('e')   = sin(2*pi*659.25*t);
notes_map('f')   = sin(2*pi*698.46*t);
notes_map('g')   = sin(2*pi*783.99*t);
notes_map('a')   = sin(2*pi*880*t);
notes_map('b')   = sin(2*pi*987.77*t);

notes_map('c_h') = sin(2*pi*523.25*t_h);
notes_map('d_h') = sin(2*pi*587.33*t_h);
notes_map('e_h') = sin(2*pi*659.25*t_h);
notes_map('f_h') = sin(2*pi*698.46*t_h);
notes_map('g_h') = sin(2*pi*783.99*t_h);
notes_map('a_h') = sin(2*pi*880*t_h);
notes_map('b_h') = sin(2*pi*987.77*t_h);

melody = "g g a g c_h b_h g g a g d_h c_h ...
          g g g_h e_h c_h b_h a_h ...
          f_h f_h e_h c_h d_h c_h ...
          g g a g c_h b_h g g a g d_h c_h";
```

```
melody = split(melody);

output = [];

for i = 1:length(melody)
    note = melody{i};
    if isKey(notes_map, note)
        output = [output notes_map(note) silence];
    end
end

sound(output, Fs);
audiowrite('mysong.wav', output, Fs);
```

Discussion

The generated audio signal successfully reproduces the selected melody. The resulting file `mysong.wav` was saved in WAV format.

Exercise 2.3: Automatic Extraction of Notes and Durations

In this exercise, a method is proposed to automatically extract the musical notes and their corresponding holding times from a given music signal. The signal is assumed to be generated according to the assumptions stated at the beginning of Part 2, namely single-tone sinusoidal notes, fixed note durations, and short silent intervals between consecutive notes.

Proposed Method

The proposed method consists of the following steps:

1. The music signal is divided into short overlapping frames.
2. Short-time energy is computed for each frame and normalized.
3. Note boundaries are detected using a relative energy threshold to separate notes from silence.
4. For each detected note segment, the dominant frequency is extracted using the Fourier transform.
5. The detected frequency is mapped to the closest musical note.

6. The duration of each note is estimated from the segment length.
7. Half-duration notes are identified based on their shorter time length and labeled with the suffix _h.

This approach combines time-domain segmentation with frequency-domain analysis, allowing both pitch and duration to be estimated reliably.

MATLAB Code

```
clc;
clear;

[y, Fs] = audioread('song_part2_1.wav');
y = y';
y = y / max(abs(y));

frame_len = round(0.02 * Fs);
hop_len   = round(0.01 * Fs);
silence_th = 0.15;

num_frames = floor((length(y) - frame_len) / hop_len);
energy = zeros(1, num_frames);

for k = 1:num_frames
    idx = (k-1)*hop_len + (1:frame_len);
    energy(k) = sum(y(idx).^2);
end

energy = energy / max(energy);
is_note = energy > silence_th;

edges = diff([0 is_note 0]);
start_idx = find(edges == 1);
end_idx   = find(edges == -1) - 1;

note_freqs = [523.25 554.37 587.33 622.25 659.25 ...
              698.46 739.99 783.99 830.61 880 932.33 987.77];
note_names = ["c","c#","d","d#","e","f","f#","g","g#","a","a#","b
   "];

detected_notes = strings(1,length(start_idx));
```

```
durations = zeros(1,length(start_idx));

for i = 1:length(start_idx)
    s = (start_idx(i)-1)*hop_len + 1;
    e = min((end_idx(i)-1)*hop_len + frame_len, length(y));
    note_sig = y(s:e);

    N = length(note_sig);
    f = (0:N-1)*(Fs/N);
    Y = abs(fft(note_sig));

    [~, idx] = max(Y(1:floor(N/2)));
    [~, nidx] = min(abs(note_freqs - f(idx)));

    detected_notes(i) = note_names(nidx);
    durations(i) = (e - s)/Fs;
end

labeled_notes = strings(1,length(detected_notes));
for i = 1:length(detected_notes)
    if durations(i) < 0.35
        labeled_notes(i) = detected_notes(i) + "_h";
    else
        labeled_notes(i) = detected_notes(i);
    end
end

for i = 1:length(labeled_notes)
    fprintf('%3d) %-4s %.2f s\n', i, labeled_notes(i), durations(
       i));
end
```

Results and Verification

The proposed method was applied to the music generated in Exercise 2.1. The extracted notes and their durations are shown below:

```
 1) d_h   0.26 s
 2) d_h   0.28 s
 3) g     0.52 s
 4) f#    0.53 s
 5) d      0.52 s
 6) d_h    0.28 s
 7) e_h    0.27 s
 8) e_h    0.28 s
 9) d_h    0.27 s
10) f#_h   0.28 s
11) e      0.52 s
12) d      0.53 s
13) e      0.52 s
14) f#     0.53 s
15) e      0.52 s
16) d_h    0.28 s
17) e_h    0.27 s
18) e_h    0.28 s
19) d_h    0.27 s
20) f#_h   0.28 s
21) d_h    0.27 s
22) e      0.53 s
23) d      0.52 s
24) e_h    0.28 s
25) d_h    0.27 s
26) f#     0.53 s
27) e      0.52 s
28) d      0.53 s
29) e_h    0.27 s
30) d_h    0.28 s
31) f#     0.52 s
32) e      0.53 s
33) d_h    0.27 s
34) d_h    0.28 s
35) e      0.52 s
36) f#_h   0.28 s
37) e_h    0.27 s
38) f#     0.53 s
39) f#_h   0.27 s
40) e_h    0.28 s
41) f#     0.52 s
42) f#     0.53 s
```

```
43) d     0.52 s
```

The extracted note sequence and durations match the original melody used in Exercise 2.1, confirming the correctness of the proposed method.