Lab 03: Signal Transformations

EE232: Signals & Systems

Muhammad Usman

February 27, 2018

Contents

1 Audio Transformation

Use of MATLAB to Flip, compress and expand an audio sequence in time

- Load the audio file using MATLAB. Play the audio file.
- Flip the audio sequence and play it back again.
- Next perform time compression on the flipped signal by a factor of 2. What does the audio sequence sound like?
- Perform time expansion by a factor of 2. What does the flipped audio sequence sound like?

1.1 MATLAB Code

```
function sound_fun( filename , factor)

[original, Fs] = audioread(filename);
flipped = flipud(original);
audiowrite('flipped.wav', flipped, Fs);
audiowrite('expand.wav', flipped, Fs / factor);
audiowrite('compress.wav', flipped, Fs * factor);

end
```

1.2 Analysis

1. *Flipped*: The original song can be listened now.
2. *Expand*: The song turns to be slow motioned.
3. *Compress*: The song turns to be high speed rap.

2 Basic Transformation of Signals

2.1 Signal Implementation

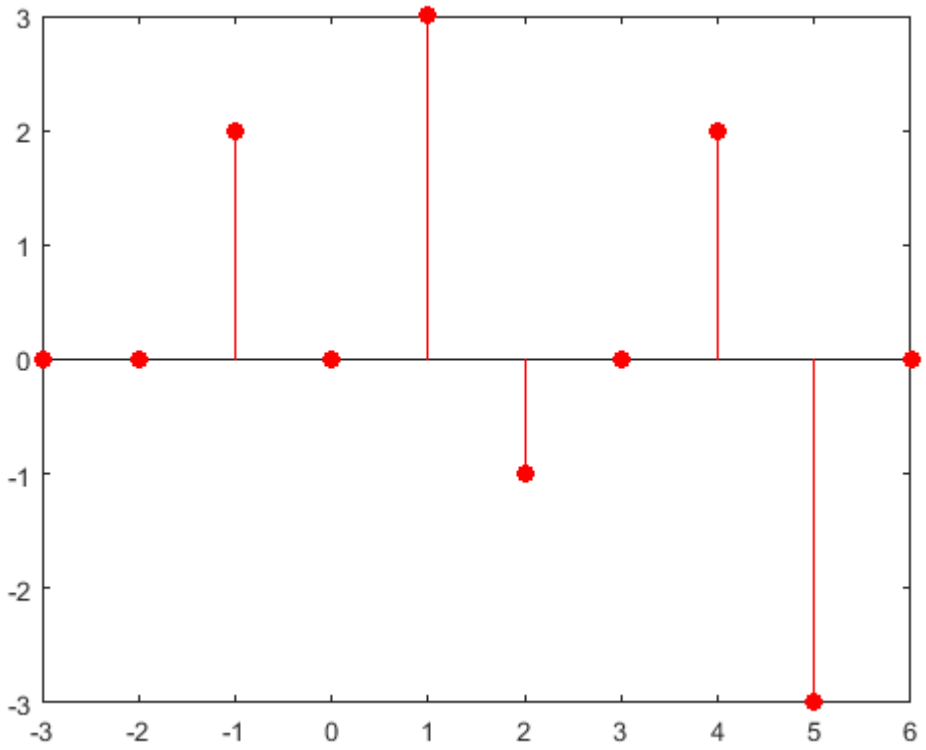
Define a MATLAB vector n, such that -3 $\leq$ n $<$ 7, representing time/sample indices of a Discrete Time Signal $x[n]$ such that

$$x[n] = \begin{cases} 2 & n = 0 \\ 3 & n = 2 \\ -1 & n = 3 \\ 2 & n = 5 \\ -3 & n = 6 \\ 0 & otherwise \end{cases}$$

2.1.1 MATLAB Code

```
n_init = -3;
n_final = 6;
n = (-3:1:6);
x = zeros(1, length(n));
x(0 - n_init) = 2;
x(2 - n_init) = 3;
x(3 - n_init) = -1;
x(5 - n_init) = 2;
x(6 - n_init) = -3;
stem(n,x,'r','filled');
```

2.1.2 MATLAB Output



2.2 Shift in Independent Axis

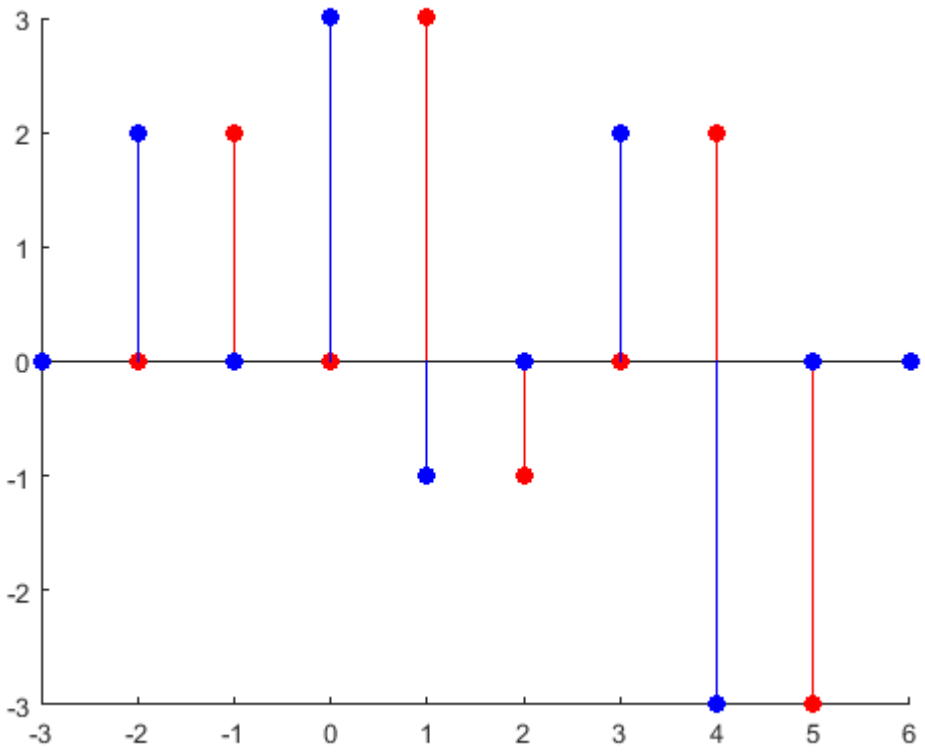
Make a generalized function that can shift the signal x[n] by delaying or advancing it by a specified amount.

2.2.1 MATLAB Code

```
function [ y ] = shift_it( x, shift )
y = zeros(1,length(x));
    for n = 0 : 1 : length(x)
        if (n - shift > 0 && n - shift < length(x))
            y(n - shift) = x(n);
        end
    end
end
```

2.2.2 MATLAB Output

```
x =

     0     0     2     0     3    -1     0     2    -3     0

>> z = shift_it(x,1)

z =

     0     2     0     3    -1     0     2    -3     0     0
```



2.3 Unit Step Function

Implement the following function in MATLAB. Your function should take k and n as input from user.

$$u[n-k] = \begin{cases} 1 & k \geq n \\ 0 & k < n \end{cases}$$

2.4 Even & Odd Constituents

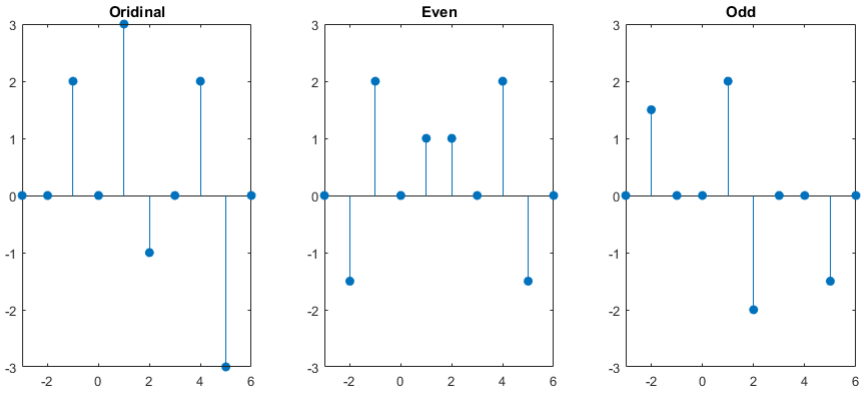
Every signal x[n] is sum of its even part and odd part. Given the signal x[n] find its even and odd part.

$$x[n] = \begin{cases} 2 & n = 0 \\ 3 & n = 2 \\ -1 & n = 3 \\ 2 & n = 5 \\ -3 & n = 6 \\ 0 & otherwise \end{cases}$$

2.4.1 MATLAB Code

```
function EorO( x, n )

x_even = 0.5 * (x + fliplr(x));
x_odd = 0.5 * (x - fliplr(x));

limit = [min(n) max(n) min(x) max(x)];

subplot(1,3,1)
stem(n,x,'filled');
axis(limit);
title('Oridinal');
subplot(1,3,2);
stem(n,x_even,'filled');
axis(limit);
title('Even');
subplot(1,3,3);
stem(n,x_odd,'filled');
axis(limit);
title('Odd');

end
```

2.4.2 MATLAB Output


Oridinal
Even
Odd

3 Image Transformation

3.1 MATLAB Code

```
punched = imread('test.jpg');
flipped = fliplr(punched);
imshow(flipped);
```

3.2 MATLAB Output

3.3 Analysis

The guy punching from his right hand has been flipped to appear being punching using his *left* hand.

4 Echo Generation

Given an audio file and the functions developed above perform a mathematical operation that will introduce echo in the song.

4.1 MATLAB Code

```
function echoIt( filename, outputFilename , echoRange )

[sound, Fs] = audioread(filename);

sound_echoed = zeros(length(sound), 1);

for echoRange = 1:echoRange
    for sumUp = echoRange:length(sound)
        if sumUp - echoRange > 0
            sound_echoed(sumUp) = sound(sumUp) + (sound(sumUp - echoRange)) / sumUp;
        end
    end
end
```

4.2 Interpretation

Here the same sound has been delayed but with a dropped amplitude to give the continuous echo effect throughout.

5 Audio Control: Forward or Reversed

Assume that you want to develop a system that loads a song and gives you control to forward or rewind the song. For this you should load the song. The total time of the song can be calculated by dividing the Total Number of Samples by Sampling Frequency

$$t_f = \frac{N_s}{f_s}$$

Next for each second, load the duration of song, play it, check if the user wants to forward or rewind the song by pressing r or f *(getkeywait(seconds))* and modify the samples being used to play the song.

5.1 MATLAB Code

```
[sound_in, Fs] = audioread('papparazi.wav');
sound_time = length(sound_in) / Fs;
second_length = length(sound_in) / sound_time;
effective_character = 'f';
previous_character = 'f';
sound_frame = 1;
while sound_frame > 0 && sound_frame < length(sound_in)

    if effective_character == 'f'
        if previous_character == 'r'
            sound_in = flipud(sound_in);
            sound_frame = length(sound_in) - sound_frame;
            previous_character = 'f';
        end
        sound(sound_in(sound_frame : sound_frame +  second_length), Fs);
        sound_frame = sound_frame + second_length;
    end

    if effective_character == 'r'
        if previous_character == 'f'
            sound_in = flipud(sound_in);
            sound_frame = length(sound_in) - sound_frame;
            previous_character = 'r';
        end
        sound(sound_in(sound_frame - second_length: sound_frame), Fs);
        sound_frame = sound_frame - second_length;
    end

    character = getkeywait(0.9 * second_length / length(sound_in) * sound_time);

    if character == 'f' || character == 'r'
        effective_character = character;
```

```
    end

end
```

5.2 Interpretation

Signal is being flipped up and down depending on the user input while most of the code revolves around receiving, and manipulating the significance of user input at a specific state and boundary checks and corrections.