Lab-2

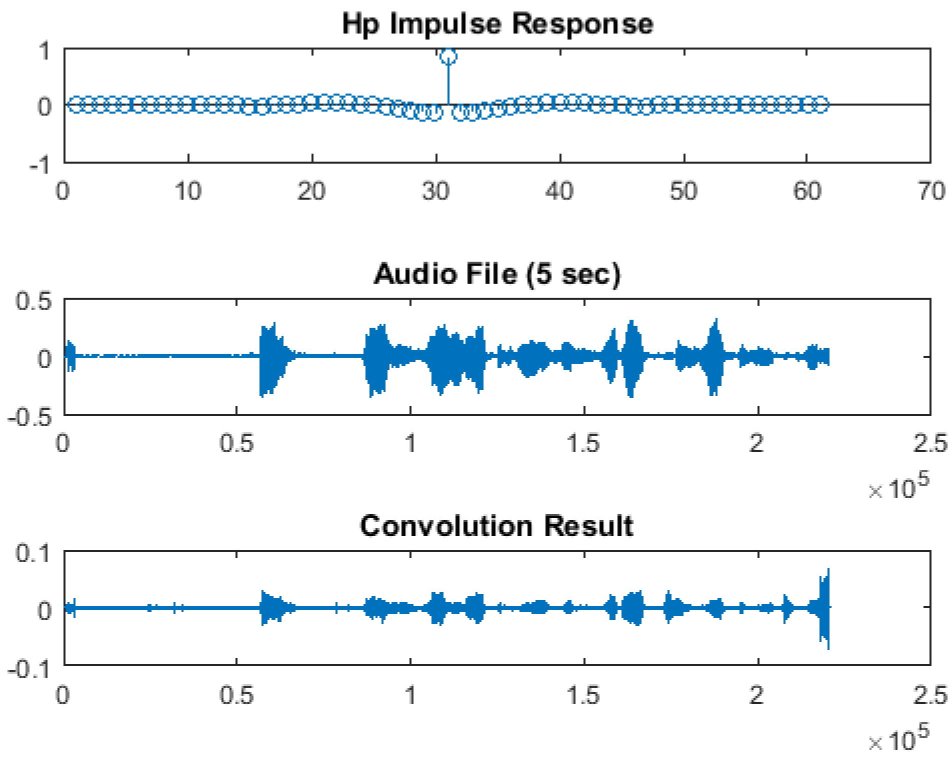
Name: Himanshu Sharma Roll Number: 1610110149 Email: hs583@snu.edu.in Instructor: Prof. Vijay Chakka

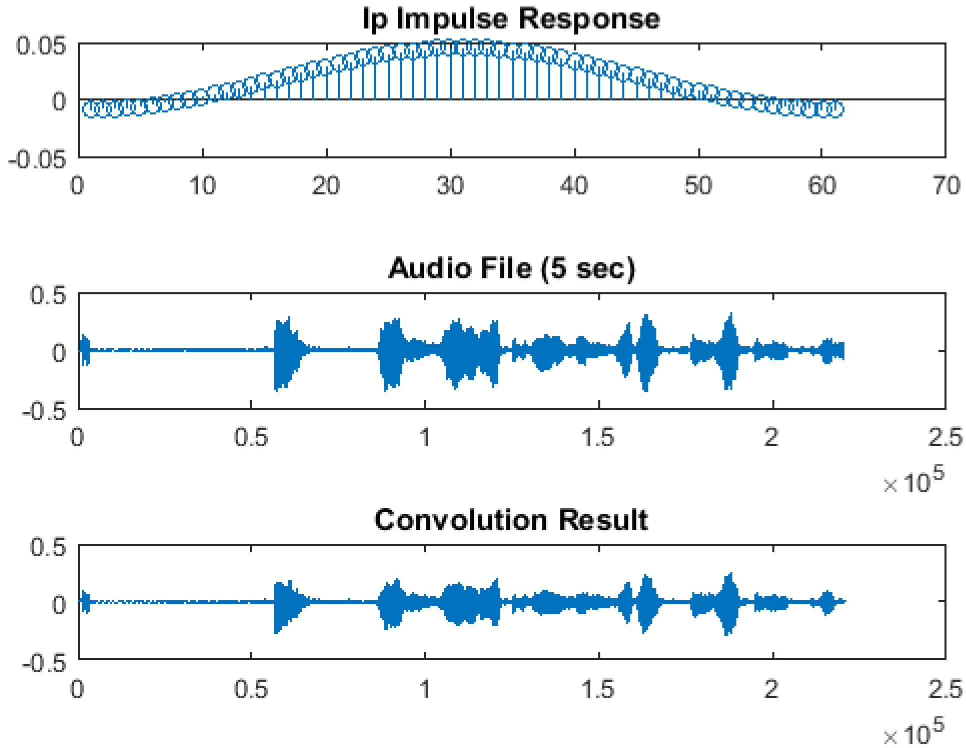
```
function a=main()
    % defining the 'main' namespace.

    % read the audio file.
    [x, fs] = audioread('audio.mp3');
    % take 5 second sample of the audio.
    v = x(1:5*fs,1);
    v = reshape(v, [], length(v));

    % loading the .mat files
    h1 = load('hpImpulseRes.mat');
    h1 = h1.h1;

    h2 = load('lpImpulseRes.mat');
    h2 = h2.h1;

    % get the convolution result. blocks = 512.
    y1 = block_convolve(v, h1, 512);
    y2 = block_convolve(v, h2, 512);

    % plot for hpimpulseres.mat
    figure;
    subplot(3,1,1);

    stem(h1);
    title('Hp Impulse Response');
    subplot(3,1,2);

    plot(v);
    title('Audio File (5 sec)');
    subplot(3,1,3);

    plot(y1);
    title('Convolution Result');

    % plot for ipimpulseres.mat
    figure;
    subplot(3,1,1);

    stem(h2);
    title('Ip Impulse Response');
    subplot(3,1,2);

    plot(v);
    title('Audio File (5 sec)');
    subplot(3,1,3);
```

```
plot(y2);
title('Convolution Result');

% define x and h for linear convolution.
x = [1,2,3,4,5,6,7];
h = [1,-1];
figure;
subplot(3,1,1);

stem(x);
title('x(n)');
subplot(3,1,2);

stem(h);
title('h(n)');
subplot(3,1,3);

stem(linear_convolution(x,h));
title('y(n)');
ylim([-8, 8]);

function y = linear_convolution(x, h)

    % first add zeros to the array which has less length.
    if length(x) < length(h)
        zeros_needed = length(h) - length(x);
        x = [x zeros(1, zeros_needed)];
    else
        zeros_needed = length(x) - length(h);
        h = [h zeros(1, zeros_needed)];
    end;
    % define the output array 'y' as array of zeros.

    y = zeros(1, length(x)+length(h)-1);

    % reverse 'x' and 'h' so that multiplication can be done
iteratively.
    x = fliplr(x);
    h = fliplr(h);

    % now iterate in 'h' element by element and multiply each of
them to 'x'
    % to create a block.
    for i = 1:length(h)
        % define current block
        block = h(i)*x;

        % add zeros to front portion.
        block = [zeros(1, i-1) block];

        % now add zeros at the end of this block.
        % the number of zeros at the end must be the length(y) -
length(block)
        block = [block zeros(1, length(y)-length(block))];
```

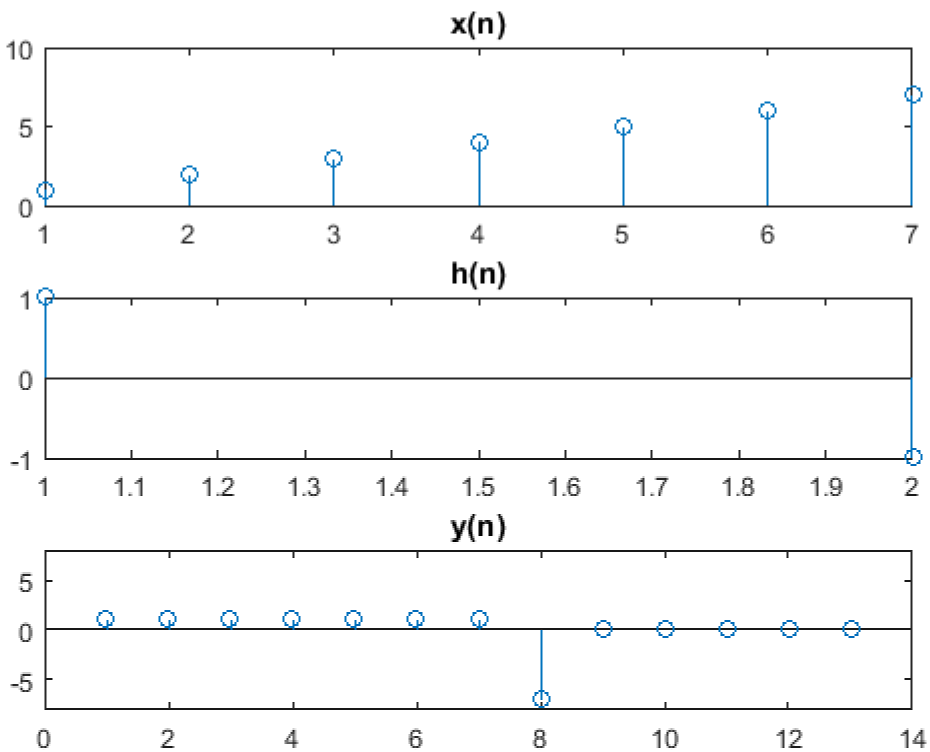
```
            % add this block to 'y'
            y = y + block;
            %disp(fliplr(block));

        end;
         % since the result is reversed, reverse it again to get
correct answer.
         y = fliplr(y);
    end;

    % define block convolution function now.
    function y = block_convolve(x, h, block_size)
        % add zeros to 'x' and 'h' to make its length a multiple of
block_size.
        x = [x zeros(1, block_size - rem(length(x), block_size))];
        h = [h zeros(1, block_size - rem(length(h), block_size))];

        % initialise a blank output array 'y' of length(x) + length(h)
- 1
        y = zeros(1, length(x)+length(h)-1);

        % decide whose length is large, x or h.
        if length(x) > length(h)
            big = x;
            small = h;
        else
            big = h;
            small = x;
        end;

        % initialise a counter variable with 1.
        counter = 1;

        % now iterate for each block in the longer array, i.e., big.
        for i = 1:block_size:length(big)
            current_block = big(i:i+block_size-1);
            %disp('Current Block:');
            %disp(current_block);

            % now convolve this block with 'small'.
            out = linear_convolution(current_block, small);

            % add zeros in front of out.
            out = [zeros(1, counter-1) out];

            % add zeros at end of out. After this the length becomes
equal to that of
            % length of 'y'.
            out = [out zeros(1, length(y) - length(out))];
            %cdisp('Output of current block');
            %disp(out);

            % add the final out block to 'y'.
```

```
                y = y + out;
                counter = counter + block_size;
            end;
        end;
    end
```





lp Impulse Response
0.05
0
-0.05
0
10
20
30
40
50
60
70
Audio File (5 sec)
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
× 10^5
Convolution Result
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
× 10^5



x(n)
10
5
0
1
2
3
4
5
6
7
h(n)
1
0
-1
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
y(n)
5
0
-5
0
2
4
6
8
10
12
14

Published with MATLAB® R2015b