Digital Signal Processing - Lab 3 Report

Shakleen Ishfar
ID: 160041029

18 July, 2019

Contents

1 First Task

1.1 Question: Task 1

Use the built-in MATLAB function conv to convolve S with H. Plot the output signal along with the original input signal and impulse response.

1.2 MATLAB Code: convSignal()

The code listed below convolutes signal s and h using the built-in function conv():

```
function output = convSignals(s, h)
    % Function convolves s with h using built in
    % conv() function of MATLab.
    %
    % Parameters:
    % s - The input signal.
    % h - The impulse response.
    %
    % Returns:
    % output - The result of convolution.
    output = conv(s, h);
end
```

1.3 Test Run: convSignal()

The following code snippet runs *convSignal(s, h)*:

```
s = [3 2 5 7 9 8 17 2];
h = [2 0 1 4];
convSig = convSignals(s, h);
disp(convSig);
% 6    4   13   28   31   43   71   48   49   70    8
plotSignals(s, h, convSig);
```

2 Second Task

2.1 Question: Task 2

Write a custom function InputSideConvolutionthat implements convolution using the Input Side Algorithm.

2.2 Theory: Input Side Algorithm

The input side algorithm shows how samples from the input signal influences many output signal samples. In the input side algorithm, we multiply each input sample with the impulse response to produce n number of signals. Here, n is the number of samples in the input signal. Then these n signals are added to produce the output signal. The appropriate samples of the same index are added with each other. The sum of all the signals is the result of convolution of input and impulse response signals.

2.3 MATLAB Code: inputSideAlgorithm()

The code listed below convolutes signal s and h using the the input side algorithm:

```
function output = inputSideAlgorithm(s, h)
    % Function convolves s with h using input
    % side algorithm.
    %
    % Parameters:
    % s - The input signal.
    % h - The impulse response.
    %
    % Returns:
    % output - The result of convolution.

    % The length of the signal produced through convolution
    % is n+m-1. Here,
    %   n = length(s)
    %   m = length(h)
    outputLength = length(s) + length(h) - 1;
    output = zeros(1, outputLength);

    % Running a loop to produce covolution output using
    % the input side algorithm technique.
    %
    % Each sample of input will be multiplied by the impulse
    % response and the final output will be a sum of all these.
    for i = 1:length(s)
        tempVec = s(1, i) * h;
        output(i: length(h)+i-1) += tempVec;
    end
end
```

2.4 Test Run: inputSideAlgorithm()

The following code snippet runs *inputSideAlgorithm(s, h)*:

```
s = [3 2 5 7 9 8 17 2];
h = [2 0 1 4];
convSig = inputSideAlgorithm(s, h);
disp(convSig);
% 6    4   13   28   31   43   71   48   49   70    8
plotSignals(s, h, convSig);
```

3 Third Task

3.1 Question: Task 3

Write another function OutputSideConvolutionthat uses the Output Side Algorithm.

3.2 Theory: Output Side Algorithm

THe output side algorithm shows how one output signal sample is influenced by many samples of different input signal samples. The general formula followed

by the output side algorithm is

$$y[i] = \sum_{j=1}^{M} h[j] * s[i - j + 1] \tag{1}$$

Here, y[i] is the i-th sample of output signal, h is the impulse response and s is the input signal.

3.3 MATLAB Code: outputSideAlgorithm()

The code listed below convolutes signal s and h using the the output side algorithm:

```
function output = outputSideAlgorithm(s, h)
    % Function convolves s with h using input
    % side algorithm.
    %
    % Parameters:
    % s - The input signal.
    % h - The impulse response.
    %
    % Returns:
    % output - The result of convolution.

    % The length of the signal produced through convolution
    % is n+m-1. Here,
    %   n = length(s)
    %   m = length(h)
    outputLength = length(s) + length(h) - 1;
    output = zeros(1, outputLength);

    % Running for loop to produce the output of convolution.
    %
    % The equation followed is
    % y[i] = sum(j = 1..M) {h[j] * s[i-j+1]}
    for i = 1:outputLength
        startIndex = min(length(s), i);
        endIndex = max(1, i-length(h)+1);
        revS = s(1, startIndex : -1 : endIndex);

        % If there aren't sufficient samples in revS
        % we add zeros to fill up the requirement
        if length(h) > length(revS)
            zeroFill = zeros(1, length(h) - length(revS));

            if endIndex != i-length(h)+1
                revS = [revS zeroFill];
            elseif startIndex != i
                revS = [zeroFill revS];
            end
        end

        output(1, i) = sum(h .* revS);
    end
end
```

3.4 Test Run: outputSideAlgorithm()

The following code snippet runs *outputSideAlgorithm(s, h)*:

```
s = [3 2 5 7 9 8 17 2];
h = [2 0 1 4];
convSig = outputSideAlgorithm(s, h);
disp(convSig);
% 6    4   13   28   31   43   71   48   49   70    8
plotSignals(s, h, convSig);
```

4 Fourth Task

4.1 Question: Task 4

Check that all 3 produce the exact same result.

4.2 MATLAB Code: outputSideAlgorithm()

The code listed below convolutes signal s and h using all 3 functions and checks if all the outputs match:

```
s = [3 2 5 7 9 8 17 2];
h = [2 0 1 4];

% Built in function conv
y1 = convSignals(s, h);

% Input side algorithm
y2 = inputSideAlgorithm(s, h);

% Output side algorithm
y3 = outputSideAlgorithm(s, h);

% y1, y2, y3 produce the result below:
% 6    4   13   28   31   43   71   48   49   70    8

% Checking if the results produced by all three methods are same.
assert(
    y1 == y2,
    "Results from conv and input side algorithm don't match!"
);
assert(
    y1 == y3,
    "Results from conv and output side algorithm don't match!"
);
assert(
    y2 == y3,
    "Results from input and output side algorithm don't match!"
);
```

5 Appendix

5.1 MATLAB Code: plotSignals()

The code listed below produces a plot for convolution functions from task 1, 2 and 3:

```
function plotSignals(s, h, c)
    % Function to plot the convolution c produced from s and h.
    %
    % Parameters:
    % s - Input signal.
    % h - Impulse response.
    % c - Result of convolution.
    set(0, "defaultaxesfontname", "Helvetica");
    hold on;
    plot(s, 'r');
    plot(h, 'b');
    plot(c, 'g');
    title('Convolve signals');
    xlabel("Time");
    ylabel("Amplitude");
    legend("Input Signal", "Impluse Response", "Convolve Signal");
    print -djpg ../Figures/Convolution.jpg
    hold off;
end
```

5.2 Produced Plot

Running plotSignals() gives the following plot for tasks 1, 2 and 3:

Figure 1: Convolution of s and h