Experiment No: 8 Date: 08/10/24

OVERLAP ADD AND OVERLAP SAVE METHOD

Aim:

Implement overlap add and overlap save method using Matlab/Scilab.

Theory:

Both the Overlap-Save and Overlap-Add methods are techniques used to compute the convolution of long signals using the Fast Fourier Transform (FFT). The direct convolution of two signals, especially when they are long, can be computationally expensive. These methods allow us to break the signals into smaller blocks and use the FFT to perform the convolution more efficiently.

Overlap-Save Method

The Overlap-Save method deals with circular convolution by discarding the parts of the signal that are corrupted by wrap-around effects. Here's how it works:

1. Block Decomposition: The input signal is divided into overlapping blocks. If the filter has length and we use blocks of length, the overlap is samples, so each block has new samples and samples from the previous block.

2. FFT and Convolution: Each block is convolved with the filter using FFT. However, because of circular convolution, the result contains artifacts due to the overlap.

3. Discard and Save: We discard the first samples from each block (the part affected by the wrap-around) and save the remaining samples. This gives us the correct linear convolution.

Overlap-Add Method

The Overlap-Add method, on the other hand, handles circular convolution by adding overlapping sections of the convolved blocks. Here's how it works:

1. Block Decomposition: The input signal is split into non-overlapping blocks of size. Each block is then zero-padded to a size of , where is the length of the filter.

2. FFT and Convolution: Each block is convolved with the filter using FFT. Since the blocks are zero-padded, the convolution produces valid linear results, but the output blocks overlap.

3. Overlap and Add: After convolution, the results of each block overlap by samples. These overlapping regions are added together to form the final output.

Program:

1. Overlap Add

```
clc;
clear all;
close all;
% User input for the input sequence
x = input('Enter the input sequence x : ');
% User input for the impulse response
h = input('Enter the impulse response h : ');
% Section length for overlap-save
L = length(h);  % Length of impulse response
% Initialization
N = length(x);
M = length(h);
% Pad input x with zeros
x_padded = [x, zeros(1, L - 1)];
% Prepare the output array
y = zeros(1, N + M +1);
% Calculate the number of sections
num_sections = (N + L - 1) / L;  % Calculate number of sections
% Process sections
for n = 0:num_sections-1
    % Determine the current section
    start_idx = n * L + 1;
    end_idx = start_idx + L - 1;
    % Ensure the section does not exceed the bounds
    x_section = x_padded(start_idx:min(end_idx, end));
    % Convolution
    conv_result = conv(x_section, h);
    % Save the results to the output
    y(start_idx:start_idx     +     length(conv_result)     -     1)
=y(start_idx:start_idx + length(conv_result) - 1) + conv_result;
end
% Trim the output to the valid part
y = y(1:N + M - 1);
% Compare with built-in convolution
y_builtin = conv(x, h);
% Display results
disp('Overlap-add convolution result:');
disp(y);
disp('Built-in convolution result:');
disp(y_builtin);
% Plotting results
figure;
subplot(2, 1, 1);
stem(y, 'filled');
title('Overlap-add Convolution Result');
grid on;
subplot(2, 1, 2);
stem(y_builtin, 'filled');
title('Built-in Convolution Result');
grid on;
```

2.Overlap Save

```
clc;
clear all;
close all;
% Input the sequences and block size
x = input("Enter 1st sequence: ");
h = input("Enter 2nd sequence: ");
```

```
N = input("Fragmented block size: ");
% Call the overlap-save function
y = ovrlsav(x, h, N);
disp("Using Overlap and Save method");
disp(y);
disp("Verification");
disp(cconv(x,h,length(x)+length(h)-1));
% Define the overlap-save method function
function y = ovrlsav(x, h, N)
    if (N < length(h))
        error("N must be greater than the length of h");
    end
    Nx = length(x);  % Length of input sequence x
    M = length(h);   % Length of filter sequence h
    M1 = M - 1;      % Length of overlap
    L = N - M1;      % Length of non-overlapping part
    % Zero-padding for input and filter sequences
    x = [zeros(1, M1), x, zeros(1, N-1)];
    h = [h, zeros(1, N - M)];
    % Number of blocks
    K = floor((Nx + M1 - 1) / L);
    % Initialize the output matrix Y
    Y = zeros(K + 1, N);

    % Perform block convolution using circular convolution
    for k = 0:K
        xk = x(k*L + 1 : k*L + N);  % Extract block of input sequence
        Y(k+1, :) = cconv(xk, h, N);  % Circular convolution
    end
    % Extract valid part from the result and concatenate
    Y = Y(:, M:N)';
    y = (Y(:))';
end
```

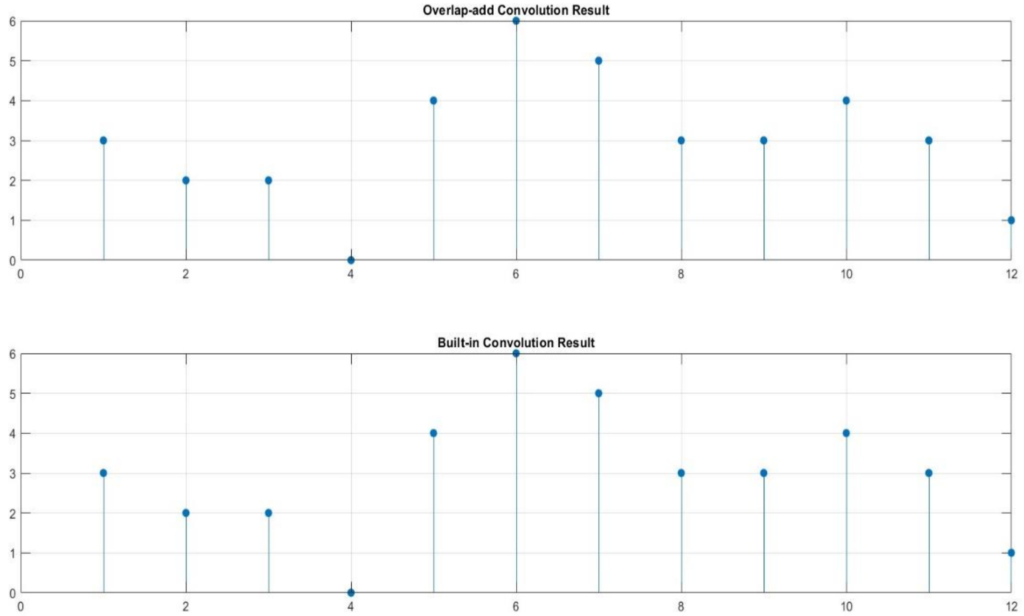
Result:

Performed Overlap Add and Overlap Save methods and verified the result.

Observation:

1. Overlap Add

```
Enter the input sequence x : [3 -1 0 1 3 2 0 1 2 1]

Enter the impulse response h : [1 1 1]

Overlap-add convolution result:

    3    2    2    0    4    6    5    3    3    4    3    1

Built-in convolution result:

    3    2    2    0    4    6    5    3    3    4    3    1
```



2.Overlap Save

```
Enter 1st sequence: [3 -1 0 1 3 2 0 1 2 1]

Enter 2nd sequence: [1 1 1]

Fragmented block size: 3

Using Overlap and Save method

    3    2    2    0    4    6    5    3    3    4    3    1

Verification

    3.0000    2.0000    2.0000         0    4.0000    6.0000    5.0000    3.0000    3.0000    4.0000
3.0000    1.0000
```