

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 M and we use blocks of length N , the overlap is M samples, so each block has $N - M + 1$ new samples and $M - 1$ 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 $M - 1$ 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 N . Each block is then zero-padded to a size of $2N - M$, where M 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 M samples. These overlapping regions are added together to form the final output.

Program:

1. Overlap Add

```
clc;
clear all;
close all;

x = input('Enter the input sequence x : ');
h = input('Enter the impulse response h : ');
L = length(h); % Length of impulse response
N = length(x);
M = length(h);
x_padded = [x, zeros(1, L - 1)];
y = zeros(1, N + M + 1);

num_sections = (N + L - 1) / L; % Calculate number of sections
for n = 0:num_sections-1
    start_idx = n * L + 1;
    end_idx = start_idx + L - 1;
    x_section = x_padded(start_idx:min(end_idx, end));

    conv_result = conv(x_section, h);
```

```

        y(start_idx:start_idx + length(conv_result) - 1)
    =y(start_idx:start_idx + length(conv_result) - 1) + conv_result;
end
y = y(1:N + M - 1);
y_builtin = conv(x, h);
% Display results
disp('Overlap-add convolution result:');
disp(y);
disp('Built-in convolution result:');
disp(y_builtin);
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;
x = input("Enter 1st sequence: ");
h = input("Enter 2nd sequence: ");

```

```

N = input("Fragmented block size: ");
y = ovrlsav(x, h, N);
disp("Using Overlap and Save method");
disp(y);
disp("Verification");
disp(cconv(x,h,length(x)+length(h)-1));
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
    x = [zeros(1, M1), x, zeros(1, N-1)];
    h = [h, zeros(1, N - M)];
    K = floor((Nx + M1 - 1) / L);
    Y = zeros(K + 1, N);

    for k = 0:K
        xk = x(k*L + 1 : k*L + N);
        Y(k+1, :) = cconv(xk, h, N);
    end
end

```

```
Y = Y(:, M:N)';  
y = (Y(:))';  
end
```

Result:

Performed Circular Convolution using a) FFT and IFFT; b) Concentric Circle method; c) Matrix method and verified result.

Observation:

1. Overlap Add

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

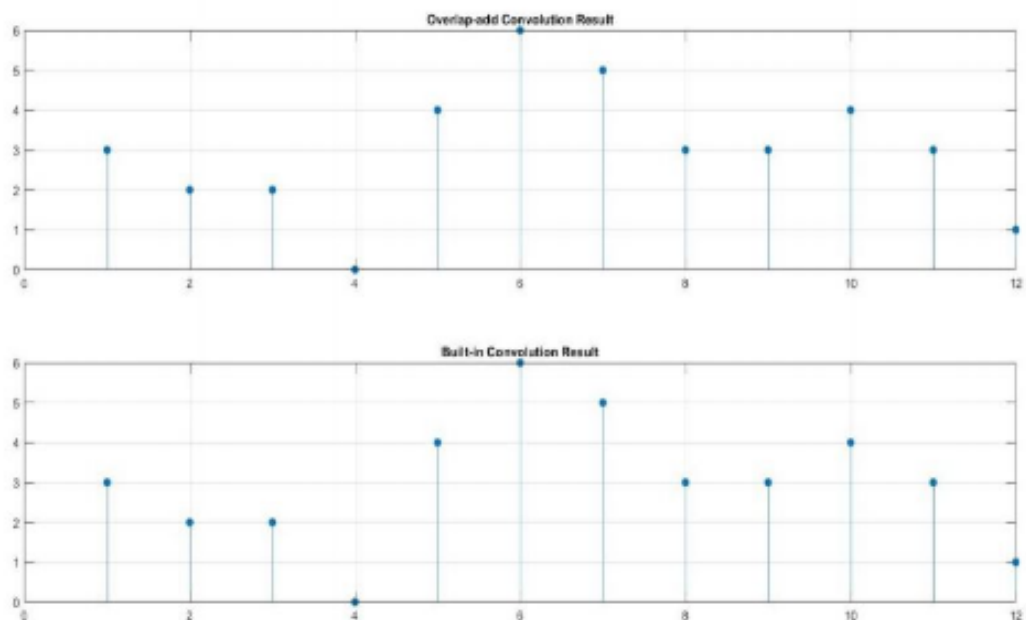
Enter the impulse response h : [1 1 1]

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