National Institute of Technology Calicut

Department of Electronics and Communication Engineering

Digital Signal Processing LAB

Week 4 Report

Date: 14–08–2025
Instructor: Dr. Sakthivel V

Submitted by:

Mummana Jagadeesh	B231113EC
Muhammed Nihal MP	B230437EC

Aim

To perform and compare:

1. Linear convolution using the Overlap-Add method.
2. Linear convolution using the Overlap-Save method.

Experiment 3A: Linear convolution using the Overlap-Add method

Theory

Overlap-Add Method (OAM) is a block-based technique for computing linear convolution of a long input sequence $x[n]$ with an FIR filter $h[n]$. Instead of processing the entire sequence at once, the signal is divided into smaller sections which are convolved separately and then recombined.

Let $x[n]$ be partitioned into blocks of length L:

$$x[n] = \sum_{r=0}^{R-1} x_r[n - rL]$$

where each block $x_r[n]$ has length L.

If $h[n]$ is of length M, then each block convolution produces a segment of length:

$$N = L + M - 1$$

Steps of the method:

1. Divide $x[n]$ into non-overlapping sections of length L.
2. For each block, compute the convolution with $h[n]$
3. Each block result has $M-1$ extra samples. These overlapping parts are added with the next block output.
4. The final result after overlap and addition equals the linear convolution $y[n]$.

MATLAB Code Listing

Code:

MATLAB Code for Linear Convolution using OAM

```
x = input('Enter input sequence x (row vector): ');
n0x = input('Enter the index of x[1] (first element of x):
    ');
```

```
h = input('Enter impulse response h (row vector): ');
n0h = input('Enter the index of h[1] (first element of h):
   ');

L = input('Enter block length L: ');

Nx = length(x);
Nh = length(h);
M  = L + Nh - 1;

num_blocks = 0;
while num_blocks*L < Nx
num_blocks = num_blocks + 1;
end

x_padded = [x, zeros(1, num_blocks*L - Nx)];
y = zeros(1, Nx+Nh-1);

for b = 0:num_blocks-1
    start_idx = b*L + 1;
    end_idx   = start_idx + L - 1;
    x_block   = x_padded(start_idx:end_idx);

    y_block = zeros(1, M);
    for n = 1:M
        s = 0;
        for k = 1:Nh
            if (n-k+1 > 0 && n-k+1 <= length(x_block))
                s = s + h(k)*x_block(n-k+1);
            end
        end
        y_block(n) = s;
    end

    y_start = start_idx;
    y_end   = min(start_idx+M-1, length(y));
    blk_end = y_end - start_idx + 1;

    y(y_start:y_end) = y(y_start:y_end) + y_block(1:blk_end
       );
end

n0y = n0x + n0h;
ny  = n0y : n0y + length(y) - 1;

yd = zeros(1, Nx+Nh-1);
```

```
for n = 1:(Nx+Nh-1)
    s = 0;
    for k = 1:Nh
        m = n-k+1;
        if (m > 0 && m <= Nx)
            s = s + h(k)*x(m);
        end
    end
    yd(n) = s;
end
nyd = n0x + n0h : n0x + n0h + length(yd) - 1;

disp('Result of convolution using Overlap-Add: ');
disp(['Indices: ', mat2str(ny)]);
disp(['Values : ', mat2str(y)]);

disp('Result of convolution using Direct Method: ');
disp(['Indices: ', mat2str(nyd)]);
disp(['Values : ', mat2str(yd)]);

subplot(4,1,1);
stem(n0x:n0x+Nx-1, x, 'filled', 'LineWidth',2);
title('Input Sequence x[n]');
xlabel('n'); ylabel('x[n]');
set(gca,'FontWeight','bold');

subplot(4,1,2);
stem(n0h:n0h+Nh-1, h, 'filled', 'LineWidth',2);
title('Impulse Response h[n]');
xlabel('n'); ylabel('h[n]');
set(gca,'FontWeight','bold');

subplot(4,1,3);
stem(ny, y, 'filled', 'LineWidth',2);
title('Output y[n] using Overlap-Add');
xlabel('n'); ylabel('y[n]');
set(gca,'FontWeight','bold');

subplot(4,1,4);
stem(nyd, yd, 'filled', 'LineWidth',2);
title('Output y[n] using Direct Convolution');
xlabel('n'); ylabel('y[n]');
set(gca,'FontWeight','bold');
```

Explanation of the Code

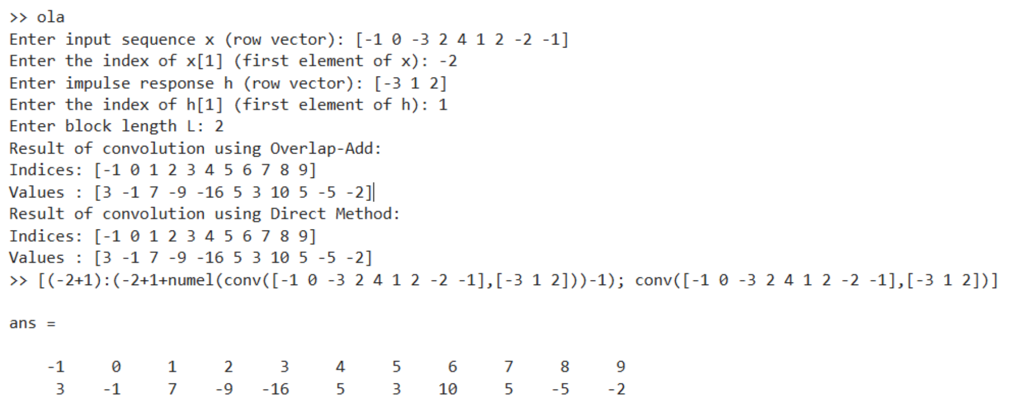
The code implements **Overlap-Add (OLA) convolution**:

- It takes an input sequence $x[n]$ and an impulse response $h[n]$, along with their respective starting indices $n0x$ and $n0h$.
- The input sequence is divided into blocks of length L to manage convolution efficiently for long signals.
- Each block is convolved with $h[n]$ using a nested loop, producing a temporary block output y_{block}.
- The block outputs are **overlapped and added** into the final output array $y[n]$. This is the core of the Overlap-Add method.
- The code also computes **direct convolution** $yd[n]$ for verification.
- Finally, it plots the input, impulse response, OLA output, and direct convolution output for comparison.

This approach is especially useful when $x[n]$ is long, as it reduces memory and computation by processing in blocks rather than convolving the entire sequence at once.

Output

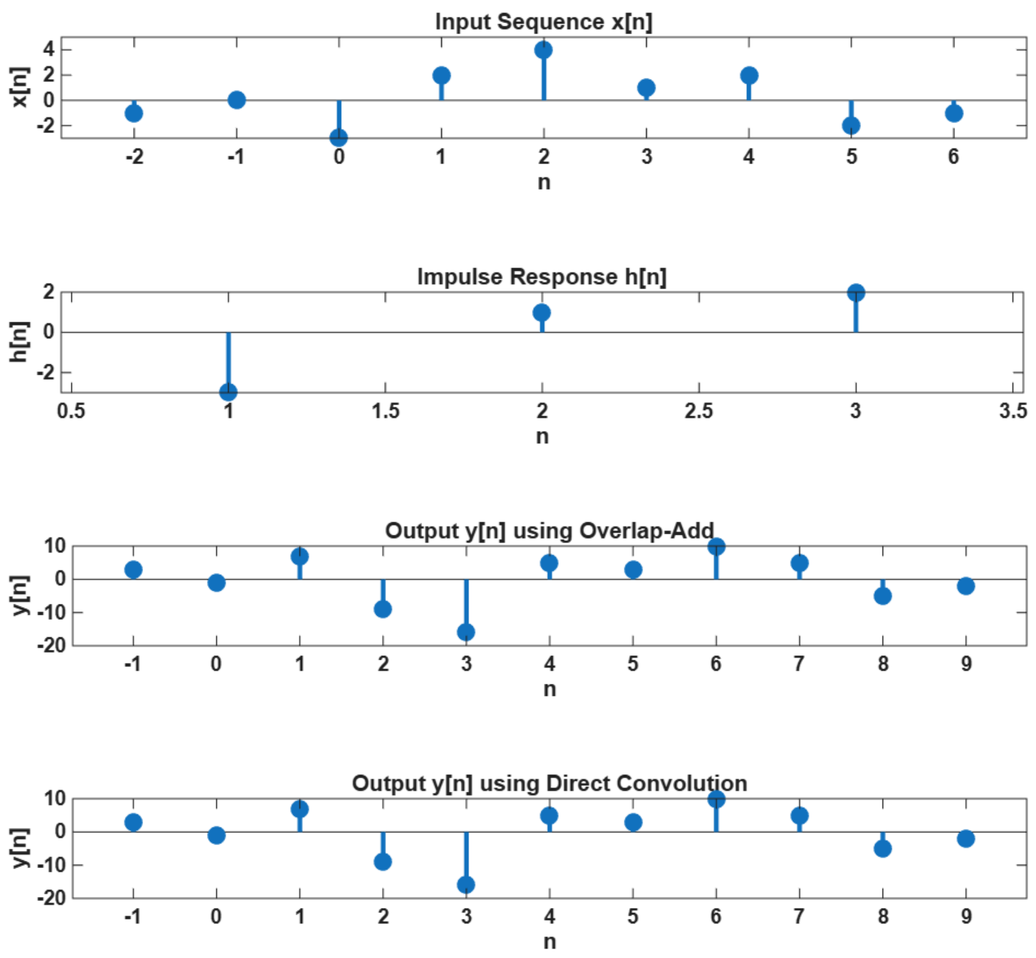
The Overlap-Add implementation was tested with the following inputs:



```
>> ola
Enter input sequence x (row vector): [-1 0 -3 2 4 1 2 -2 -1]
Enter the index of x[1] (first element of x): -2
Enter impulse response h (row vector): [-3 1 2]
Enter the index of h[1] (first element of h): 1
Enter block length L: 2
Result of convolution using Overlap-Add:
Indices: [-1 0 1 2 3 4 5 6 7 8 9]
Values : [3 -1 7 -9 -16 5 3 10 5 -5 -2]
Result of convolution using Direct Method:
Indices: [-1 0 1 2 3 4 5 6 7 8 9]
Values : [3 -1 7 -9 -16 5 3 10 5 -5 -2]
>> [(-2+1):(-2+1+numel(conv([-1 0 -3 2 4 1 2 -2 -1],[-3 1 2]))-1); conv([-1 0 -3 2 4 1 2 -2 -1],[-3 1 2])]

ans =

    -1     0     1     2     3     4     5     6     7     8     9
     3    -1     7    -9   -16     5     3    10     5    -5    -2
```

Figure 1: Command Window Output

Output Plot:



Experiment 3B: Linear convolution using the Overlap-Save method

Theory

Overlap-Save Method (OSM) is a block-based technique for computing linear convolution of a long input sequence $x[n]$ with an FIR filter $h[n]$. Instead of convolving the entire sequence at once, the signal is processed in overlapping blocks, and only the non-corrupted parts of each block are retained.

Let $x[n]$ be partitioned into overlapping sections of length $N = L + M - 1$, where L is the useful output block length and M is the length of the impulse response $h[n]$.

If $h[n]$ is of length M, then each block convolution produces a segment of length:

$$N = L + M - 1$$

Steps of the method:

1. Prepend $(M - 1)$ zeros to the input sequence $x[n]$.
2. Divide the padded sequence into overlapping sections of length N, where each block overlaps the previous one by $(M - 1)$ samples.
3. For each block, compute the convolution with $h[n]$.
4. Discard the first $(M - 1)$ samples of each block output (since they are corrupted by overlap).
5. Concatenate the remaining L samples of each block output to form the final result $y[n]$.

The output obtained is identical to the linear convolution of $x[n]$ and $h[n]$, but computed efficiently in a block-by-block manner.

MATLAB Code Listing

Code:

MATLAB Code for Linear Convolution using OSM

```matlab
x = input('Enter input sequence x (row vector): ');
n0x = input('Enter the index of x[1] (first element of x):
    ');

h = input('Enter impulse response h (row vector): ');
n0h = input('Enter the index of h[1] (first element of h):
    ');

L = input('Enter block length L: ');
```

```
Nx = length(x);
Nh = length(h);
M  = L + Nh - 1;

num_blocks = 0;
while num_blocks*L < Nx + Nh - 1
    num_blocks = num_blocks + 1;
end

x_padded = [zeros(1,Nh-1), x];
x_padded = [x_padded, zeros(1, num_blocks*L + Nh - 1 -
   length(x_padded))];

y = [];

for b = 0:num_blocks-1
    start_idx = b*L + 1;
    end_idx   = start_idx + M - 1;
    x_block   = x_padded(start_idx:end_idx);

    y_block = zeros(1, M);
    for n = 1:M
        s = 0;
        for k = 1:Nh
            if (n-k+1 > 0) && (n-k+1 <= length(x_block))
                s = s + h(k)*x_block(n-k+1);
            end
        end
        y_block(n) = s;
    end

    y = [y, y_block(Nh:end)];
end

y = y(1:Nx+Nh-1);

n0y = n0x + n0h;
ny  = n0y : n0y + length(y) - 1;

yd = zeros(1, Nx+Nh-1);
for n = 1:(Nx+Nh-1)
    s = 0;
    for k = 1:Nh
        m = n-k+1;
        if (m > 0 && m <= Nx)
            s = s + h(k)*x(m);
            end
        end
        yd(n) = s;
end
nyd = n0x + n0h : n0x + n0h + length(yd) - 1;

disp('Result of convolution using Overlap-Save: ');
disp(['Indices: ', mat2str(ny)]);
disp(['Values : ', mat2str(y)]);

disp('Result of convolution using Direct Method: ');
disp(['Indices: ', mat2str(nyd)]);
disp(['Values : ', mat2str(yd)]);

subplot(4,1,1);
stem(n0x:n0x+Nx-1, x, 'filled', 'LineWidth',2);
title('Input Sequence x[n]');
xlabel('n'); ylabel('x[n]');
set(gca,'FontWeight','bold');

subplot(4,1,2);
stem(n0h:n0h+Nh-1, h, 'filled', 'LineWidth',2);
title('Impulse Response h[n]');
xlabel('n'); ylabel('h[n]');
set(gca,'FontWeight','bold');

subplot(4,1,3);
stem(ny, y, 'filled', 'LineWidth',2);
title('Output y[n] using Overlap-Save');
xlabel('n'); ylabel('y[n]');
set(gca,'FontWeight','bold');

subplot(4,1,4);
stem(nyd, yd, 'filled', 'LineWidth',2);
title('Output y[n] using Direct Convolution');
xlabel('n'); ylabel('y[n]');
set(gca,'FontWeight','bold');
```

Explanation of the Code

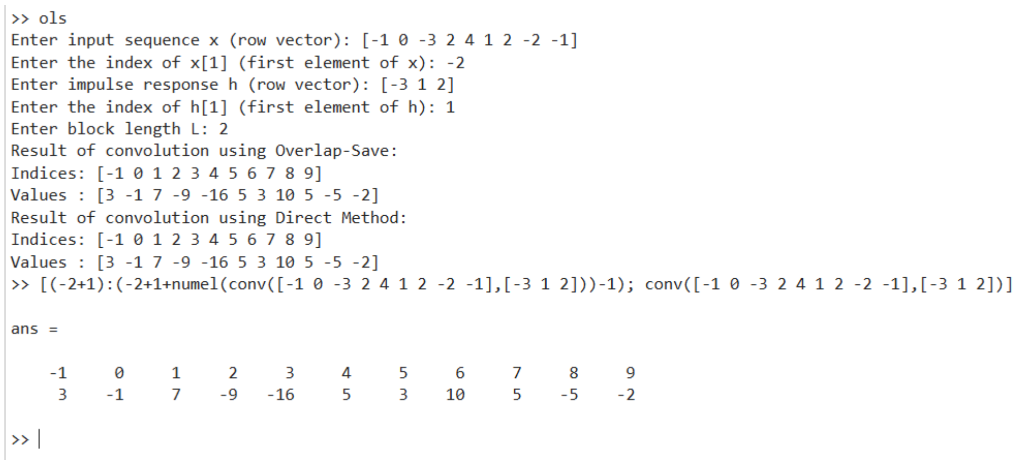
The code implements **Overlap-Save (OLS) convolution**:

- It takes an input sequence $x[n]$ and an impulse response $h[n]$, along with their starting indices $n0x$ and $n0h$.
- The input is **padded at the beginning** with $Nh-1$ zeros to handle initial overlap, and further zero-padded to fit an integer number of blocks.
- The input is divided into blocks of length $M = L + Nh - 1$. Each block overlaps the previous one by $Nh - 1$ samples.
- Each block is convolved with $h[n]$ using nested loops, producing y_{block}. Only the **valid part of each block** (excluding the initial overlapped samples) is saved.
- The final output $y[n]$ is constructed by concatenating these valid block outputs.
- Direct convolution $yd[n]$ is also computed for comparison.
- Finally, the input, impulse response, OLS output, and direct convolution output are plotted.

This method is efficient for long signals because it avoids redundant computations by discarding overlapped parts that would otherwise be recomputed, unlike in the Overlap-Add method.

Output

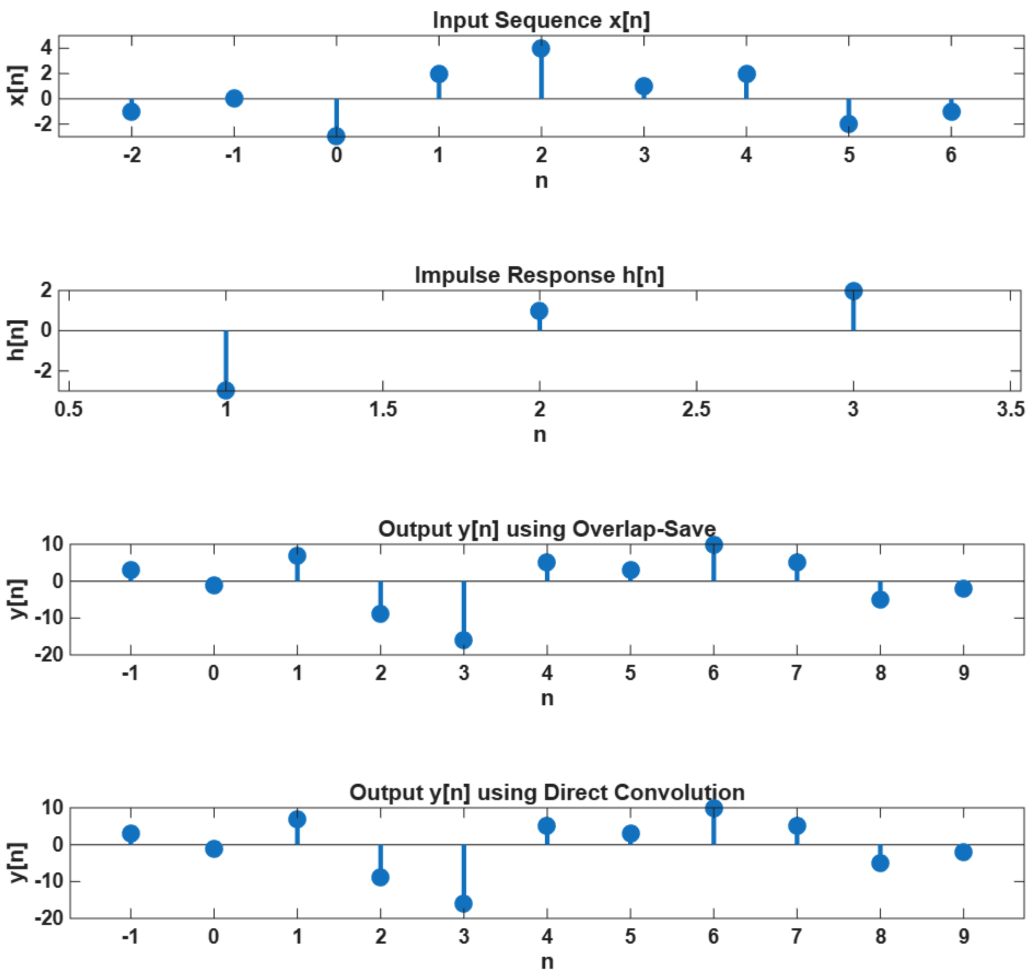
The Overlap-Save implementation was tested with the following inputs:



```
>> ols
Enter input sequence x (row vector): [-1 0 -3 2 4 1 2 -2 -1]
Enter the index of x[1] (first element of x): -2
Enter impulse response h (row vector): [-3 1 2]
Enter the index of h[1] (first element of h): 1
Enter block length L: 2
Result of convolution using Overlap-Save:
Indices: [-1 0 1 2 3 4 5 6 7 8 9]
Values : [3 -1 7 -9 -16 5 3 10 5 -5 -2]
Result of convolution using Direct Method:
Indices: [-1 0 1 2 3 4 5 6 7 8 9]
Values : [3 -1 7 -9 -16 5 3 10 5 -5 -2]
>> [(-2+1):(-2+1+numel(conv([-1 0 -3 2 4 1 2 -2 -1],[-3 1 2]))-1); conv([-1 0 -3 2 4 1 2 -2 -1],[-3 1 2])]

ans =

    -1     0     1     2     3     4     5     6     7     8     9
     3    -1     7    -9   -16     5     3    10     5    -5    -2

>> |
```

Figure 2: Command Window Output

Output Plot:


Input Sequence x[n]
x[n]
n
Impulse Response h[n]
h[n]
n
Output y[n] using Overlap-Save
y[n]
n
Output y[n] using Direct Convolution
y[n]
n