Overlap - add method

Introduction:

The Overlap-Add (OLA) method is a digital signal processing technique used to efficiently filter long signals with a finite impulse response (FIR) filter. This method involves breaking the long input signal into smaller segments, zero-padding each segment to prevent circular convolution, convolving each padded segment with the FIR filter, and then overlapping and adding the convolved segments to reconstruct the final output signal. By processing smaller segments individually and then combining them, the OLA method reduces the computational complexity associated with directly convolving a long signal, making it an efficient approach for filtering in DSP.

Methodology:

The Overlap-Add (OLA) method efficiently filters long signals with a finite impulse response (FIR) filter by segmenting the input signal and processing each segment individually. Here's a step-by-step methodology:

1. **Segmenting the Input Signal:**
   - Divide the long input signal x(n)into smaller, non-overlapping segments of length M.
   - Each segment xi(n)can be represented as:
     $$x(n)=i\sum xi(n-iM)$$

2. **Zero-Padding:**
   - Zero-pad each segment to a length N=M+L−1, where L is the length of the FIR filter h(n). This prevents circular convolution.
   - The zero-padded segment xi′(n) can be written as:
     $$xi'(n)= [xi(0),xi(1),\ldots,xi(M-1),0,\ldots,0]$$

3. **Convolution of Segments:**
   - Convolve each zero-padded segment xi′(n) with the FIR filter h(n). This step computes the linear convolution of the segment with the filter.
   - The convolved output yi(n) for each segment is:
     $$yi(n)=xi'(n)*h(n)$$

4. **Overlap and Add:**
   - Combine the convolved segments by overlapping and adding the appropriate parts to reconstruct the final output signal y(n).
   - Specifically, each convolved segment yi(n) is added at the correct position in the output signal to account for the offset introduced by segmenting:
     $$y(n)=i\sum yi(n-iM)$$

Calculations:

$x(n) = \{1, 2, -1, 2, 3, -2, -3, -1, 1, 1, 2, -1\}$

$h(n) = \{1, 2\}$

Step 1:

L_x = length of $x(n)$ = 12

M = length of $h(n)$ = 2

$N = 2^M = 2^2 = 4$

$L = N - M + 1$

$= 4 - 2 + 1 = 3$

Step 2:

$x_1(n) = \{\underbrace{1, 2, -1}_{L=3}, \underbrace{0}_{M-1 = (2-1) = 1}\}$

[zero padding for 1st sequence]

Ill 4

$x_2(n) = \{2, 3, -2, 0\}$

$x_3(n) = \{-3, -1, 1, 0\}$

$x_4(n) = \{1, 2, -1, 0\}$

Step 3:

$h(n) = \{1, 2, \underbrace{0, 0}_{L+1 = (3+1) = 4}\}$

(zero padding) for 2nd sequence

Step 4: Matrix

$$y(n)]_{N\times 1} = [x(n)]_{N\times N}[h(n)]_{N\times 1}$$

$$y_1(n) = \begin{bmatrix} 1 & 0 & -1 & 2 \\ 2 & 1 & 0 & -1 \\ -1 & 2 & 1 & 0 \\ 0 & -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ -2 \end{bmatrix}$$

$$y_2(n) = \begin{bmatrix} 2 & 0 & -2 & 3 \\ 3 & 2 & 0 & -2 \\ -2 & 3 & 2 & 0 \\ 0 & -2 & 3 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 4 \\ -4 \end{bmatrix}$$

$$y_3(n) = \begin{bmatrix} -3 & 0 & 1 & -1 \\ -1 & -3 & 0 & 1 \\ 1 & -1 & -3 & 0 \\ 0 & 1 & -1 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ -7 \\ -1 \\ 2 \end{bmatrix}$$

$$y_4(n) = \begin{bmatrix} 1 & 0 & -1 & 2 \\ 2 & 1 & 0 & -1 \\ -1 & 2 & 1 & 0 \\ 0 & -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ -2 \end{bmatrix}$$

Step 5:

Let, y_5: $\begin{Bmatrix} 1 & 4 & 3 & -2 \\ 2 & 7 & 4 & -4 \\ -3 & -7 & -1 & 2 \\ 1 & 4 & 3 & 2 \end{Bmatrix}$ (merging after interchanging rows to colums of y_1, y_2, y_3 & y_4)

Now;

$$z_1 = \begin{Bmatrix} 1 & 0 & -7 & 3 \\ 4 & 7 & -7 & 4 \\ 3 & 4 & -1 & 3 \end{Bmatrix}$$ (considering L elements of y_1, y_2, y_3 & 4 & overlapping by adding M-1 term/s)

Step 6:

$y_1 = \{1\ 4\ 3\ 0\ 7\ 4\ -7\ -7\ -1\ 3\ 4\ 3\}$.

Step 7: Constructing final output:

$y = [y_1\ u(K, (M:N))]$;

ie y_1 + remaining part of last block that wasn't included in 'y_5'

$\therefore y = \{1\ 4\ 3\ 0\ 7\ 4\ -7\ -7\ -1\ 3\ 4\ 3\ 4\ 3\ 2\}$

Code:

```
close all;
clear all;
x=input('Enter First Sequence x[n]= ');
h=input('Enter Second Sequence h[n]= ');
N=input('Enter length of each block N = ');
Lx=length(x);
M=length(h);
L=N-M+1;
K=ceil(Lx/L)
R=rem(Lx,L);

%Padding zeros to input sequences to make length equal to
N
if R>0
x=[x zeros(1,L-R)]
end
h=[h zeros(1,N-M)]

%Initializing the Output
y=zeros(N,K);

%Padding zeros to Input sequence at the end of the
%sequence
z=zeros(1,M-1);

%To perform Circular Convolution of two input sequences
for i=0:K-1
Xn=x(L*i+1:L*i+L);
Xi=[Xn z];
u(i+1,:)=cconv(Xi,h,N) %u(i+1,:)=C_Conv(Xi(i,:),h);
end
Y=u';
M1=M-1;
p=L+M1;
for i=1:K-1
u(i+1,1:M-1)=u(i,p-M1+1:p)+u(i+1,1:M-1);
end
z1=u(:,1:L)'
y1=(z1(:))'
y=[y1 u(K,(M:N))]

% Ploting the Input Sequences
subplot (2,2,1);
stem(x);
title('First Sequence x[n]');
xlabel ('Samples');
ylabel ('Amplitude');
subplot (2,2,2);
```

```
stem(h);
title('Second Sequence h[n]');
xlabel ('Samples');
ylabel ('Amplitude');

%Plotting of the Convolved Signal
subplot (2,2,3:4);
stem(y);
title ('Convolved Signal');
xlabel ('Samples');
ylabel ('Amplitude');
```

Input: Enter First Sequence x[n]= [1 2 -1 2 3 -2 -3 -1 1 1 2 -1]
Enter Second Sequence h[n]= [1 2]
Enter length of each block N = 4

Output : K = 4

N = 4

h = 1 2 0 0

u = 1 4 3 -2

u = 1 4 3 -2
2 7 4 -4

u = 1 4 3 -2
2 7 4 -4
-3 -7 -1 2

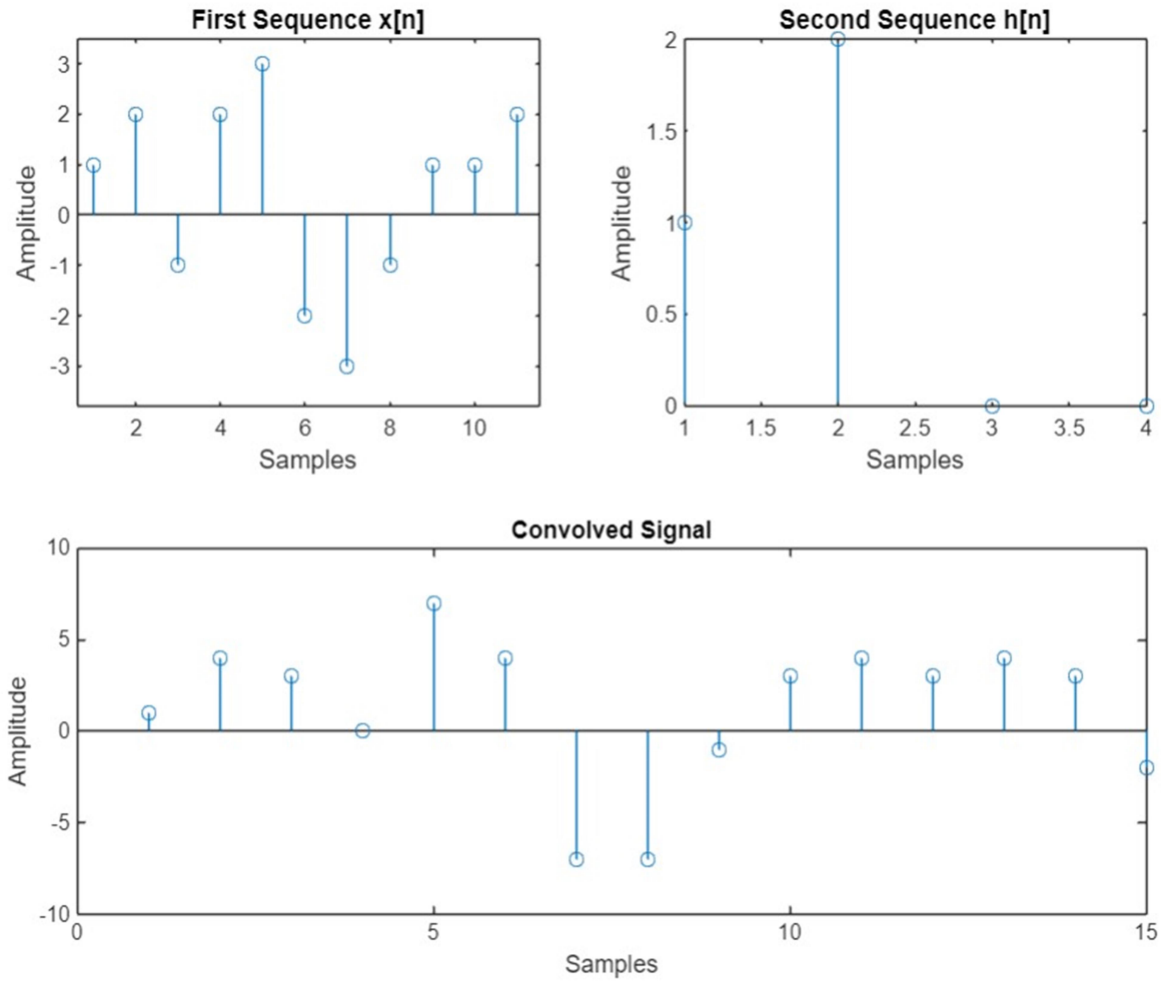
u = 1 4 3 -2
2 7 4 -4
-3 -7 -1 2
1 4 3 -2

z1 = 1 0 -7 3
4 7 -7 4
3 4 -1 3

y1 = 1 4 3 0 7 4 -7 -7 -1 3 4 3

y = 1 4 3 0 7 4 -7 -7 -1 3 4 3 4 3 -2

Graph :


First Sequence x[n]
Amplitude
Samples
Second Sequence h[n]
Amplitude
Samples
Convolved Signal
Amplitude
Samples

Applications:

- **Speech Processing**: Enhances speech quality by reducing noise. Facilitates voice conversion by applying specific filters.
- **Image Processing**: Filters and restores large images by processing smaller blocks. Manages computational complexity efficiently.
- **Biomedical Signal Processing**: Filters noise from long ECG and EEG signals. Enhances medical images for better diagnosis.
- **Telecommunications**: Applies to channel equalization, improving signal quality. Used in modulation and demodulation processes for efficient signal filtering.
- **Radar and Sonar Systems**: Helps in signal detection and processing. Enhances accuracy in identifying objects.
- **Data Compression**: Utilized in audio and video compression algorithms like MP3 and MPEG. Efficiently applies filters during encoding and decoding.

References:

- Oppenheim, A. V., & Schafer, R. W. (2009). Discrete-Time Signal Processing (3rd ed.) Pearson.
- Proakis, J. G., & Manolakis, D. G. (2006). Digital Signal Processing: Principles, Algorithms, and Applications. (4th ed.). Pearson.
- Allen, J. B., & Rabiner, L. R. (1977). "A unified approach to short-time Fourier analysis and synthesis," Proceedings of the IEEE, 65(11), 1558-1564.in