Application of Fast Fourier Transform (FFT) Algorithm in Finite Impulse Response (FIR) Linear Filtering Using MATLAB

[1]S.Arunachalam, [2]S.M.Khairnar, [3]B.S.Desale
[1]Department of Mathematics, Rizvi College of Arts, Science and Commerce, Mumbai and Research Scholar, North Maharashtra University, Jalgaon, India
[2]Department of Mathematics, MIT Academy of Engineering, Pune, India.
[3]Department of Mathematics, University of Mumbai, Mumbai, India.

Abstract

The convolution operation involved in linear filtering is computationally involved task. Since the Discrete Fourier Transform (DFT) provides a discrete frequency representation of a finite duration sequence in the frequency domain, it is interesting to explore its use as a computational tool for linear system analysis and especially for linear filtering. In this paper, we propose how the DFT can be used to perform linear filtering in the frequency domain using MATLAB program. The frequency domain approach based on the DFT is computationally more efficient than time domain convolution due to the existence of the efficient Fast Fourier Transform (FFT) algorithms for computing DFT. We have implemented MATLAB Program to obtain the output results and the graph of input and output signal for two fast convolution methods, namely, overlap-add and overlap-save algorithms.

Keywords: *DFT, FFT, Linear Filtering, Fast convolution, Finite Impulse Response (FIR), Overlap-add algorithm, Overlap-save algorithm.*

1. Introduction

FFT is a fast algorithm for execution of DFT. The multiplication of two DFTs is equivalently a circular convolution of the two sequences. We can make use of DFT for computation of convolution as the circular convolution in time domain is equivalently a multiplication in the DFT domain using a property of convolution. We need to convert the linear convolution to a circular convolution and then make use of FFT algorithm to reduce the number of computation for filtering. The procedure to convert the linear convolution to a circular convolution is increase length of each sequence by appending zeros so that it is equal to the length of the resulting convolved sequence. Then circularly convolve the resulting appended sequences. The result is same as that of linear convolution. To make use of FFT algorithm, the length of the sequences must be some power of 2.

In 1965, Cooley and Tukey showed the procedure to substantially reduce the amount of computation in the DFT [1]. The FFT introduced in 1976 by Winogard [6] stands out for achieving a new theoretical reduction in the order of the multiplicative complexity. The derivation of fast running FIR algorithms was introduced by Z.J.Mau and P.Duhamel [7]. J.W.Cooley et al. used FORTRAN program to compute DFT by FFT method [4].

Linear filtering and Fourier transforms are among the most fundamental operations in digital signal processing. Direct computation of both convolution and DFT requires on the order of N^2 operations where N is the filter length or transform size. The break-through of the Cooley-Tukey FFT comes from the fact that it brings the complexity down to an order of $N\log_2 N$ operations [2, 3, 5]. Although the DFT is computable transform, the straightforward implementation is very inefficient, especially the sequence length N is very large. In this paper, we implement MATLAB program to perform the computation of the linear convolution. MATLAB is a high-level language that enables to perform computationally intensive tasks faster than with traditional programming languages such as C, C^{++} and FORTRAN.

One of the first applications of the Cooley-Tukey FFT algorithm was to implement convolution faster than the usual direct method [9, 10, 11, 15]. Finite Impulse Response (FIR) digital filters and convolution are defined by

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \qquad (1)$$

where $x(n)$ is a length L sequence of numbers considered to be the input signal, $h(n)$ is a length M sequence of numbers considered to be the filter coefficients, and $y(n)$ is the filtered output. Equation (1) shows that the output signal $y(n)$ must be a length of L + M – 1. Our aim is calculate this convolution or filtering faster than directly implementing (1).The input sequence $x(n)$ is frequently a very long sequence when real-time signal processing is done. This is true in some real-time signal processing applications concerned with signal monitoring and analysis. Since linear filtering performed via the DFT involves operation on a block of data, a long input signal sequence must be segmented to fixed size block prior to processing. Since the filtering is linear, successive blocks can be processed one at a time via the DFT, and the output blocks are fitted together to form the overall output signal sequence.

We now describe two fast convolution or fast filtering methods using FFT, namely, overlap-add method and overlap-save method. For both these methods, we have to make use of circular convolution. If the size of each block of data is L and the size of filter length is M then the convolved sequence length is equal to L + M – 1. We have to append zeros at the end of each data block so as to make the processing block size of L + M – 1. For linear filtering, the filtered output will have a size of L + M – 1. If we use small size of the block for FFT computation such that its length is less than the number of samples in the data then there will be aliasing in time domain according to the sampling theorem in frequency domain. FFT size of L + M – 1 is equal to the number of samples in the convolved sequence in time domain [17, 18, 19]. This paper is organized as follows. Section 2 gives the overlap-add method and algorithm. Section 3 describes MATLAB program for overlap-add algorithm. Section 4 describes the overlap-save algorithm. Section 5 gives the MATLAB program for overlap-save algorithm. Section 6 gives the MATLAB program and the graph of plot of input and output signal and the conclusion is given in Section 7.

2. Overlap-add method

Consider a long input data sequence $x(n)$ of length X. Let us divide the input sequence into blocks of size L. Let the Finite Impulse Response (FIR) filter be of length M. The result of linear convolution of the input sequence of the length L with the filter of length M is a sequence of length L + M – 1 [8, 16, 20]. To execute this linear convolution using circular convolution, we have to convert linear convolution into circular convolution by appending both the sequences by extra zeros until the length of both the sequences is L + M – 1. We can use FFT to execute the circular convolution and the length of FFT as N ≥ L + M – 1. Since each data block is terminated with M – 1 zeros, the last M – 1 points from each output block must be overlapped and added to the M – 1 points of the succeeding block. This overlapping and adding gives the output sequence [12, 13, 14]. The segmentation of the input data into blocks and the fitting of output data blocks to form the output sequence are illustrated in Fig.2.

2.1 Overlap-add Algorithm

Step 1: Take the L samples of data sequence $x(n)$. Append M – 1 extra zeros to this block of data so that its length is L + M – 1.

Step 2: Append L – 1 extra zeros to the FIR filter so that its length is L + M – 1.

Step 3: Convolve the two sequences circularly using FFT as shown in Fig. 1 to obtain the output as L + M – 1.

Step 4: Repeat steps 1 to 3 for all the blocks.

Step 5: Accumulate the results by adding the overlapped samples of the convolved output for

successive blocks as shown in Fig.2.

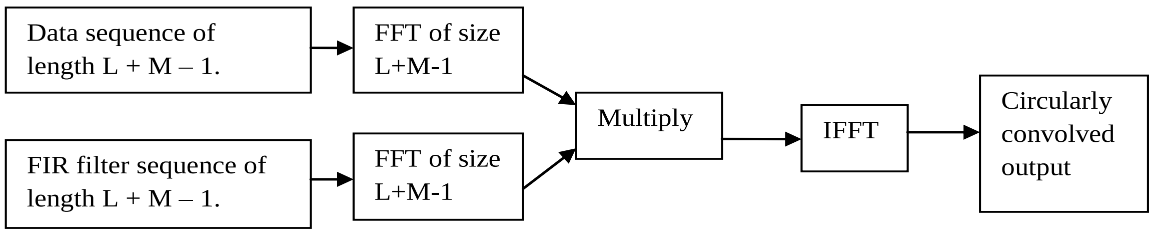
Data sequence of length L + M – 1.
FIR filter sequence of length L + M – 1.
FFT of size L+M-1
FFT of size L+M-1
Multiply
IFFT
Circularly convolved output

Fig.1 Block schematic for circular convolution using FFT.

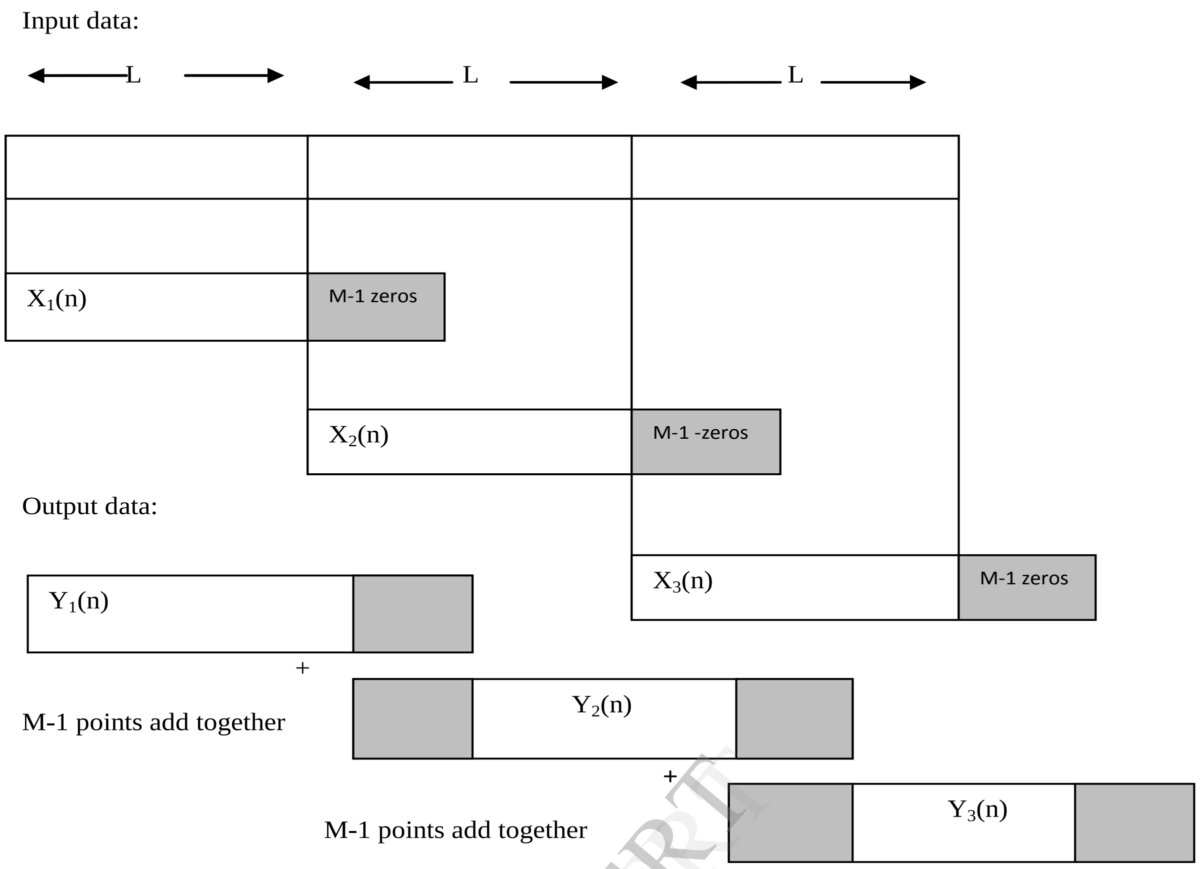

Fig.2 Linear FIR filtering by the overlap-add method.

For example, let the input sequence be $x(n)$ = [1 2 3 4 5 1 2 3 4 5 1 2 3 4 5]. Let the FIR filter sequence be $h(n)$ = [3 2 1 1]. Let us calculate the convolution using overlap-add method. We will use block size for the data as L = 5 and M = 4. We have to append three zeros to the data block and 4 zeros to $h(n)$. Therefore, the length of both the sequences is L + M – 1 = 5 + 4 – 1 = 8. When we use FFT, N is selected as some exponential of 2. The data blocks will be

$$x_1(n) = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 0\ \ 0\ \ 0]$$
$$x_2(n) = [\,1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 0\ \ 0\ \ 0]$$
$$x_3(n) = [\,1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 0\ \ 0\ \ 0]$$
$$h(n) = [\,3\ \ 2\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\,]$$

3. MATLAB Program for Overlap-Add Algorithm

The result of convolution of data blocks is obtained by using a MATLAB program. A MATLAB program and the output sequence is as follows:

```
>> %overlap add algorithm
>> clear all;
>> a=[1 2 3 4 5 1 2 3 4 5 1 2 3 4 5];
>> b=[3 2 1 1 0 0 0 0];
>> c=conv(a,b);
>> disp(c);
   3   8   14   21   28   20   17   19   21   28   20   17   19   21   28   17    9    5    0    0    0    0
>> b1=fft(b,8);
>> a1=[1 2 3 4 5 0 0 0];
>> a11=fft(a1,8);
>> for i=1:8,
     a111(i)= a11(i)*b1(i);
end
>> c1=ifft(a111,8);
>> disp(c1);
   3.0000    8.0000   14.0000   21.0000   28.0000   17.0000    9.0000    5.0000
>> a2=[1 2 3 4 5 0 0 0];
>> a22=fft(a2,8);
>> for i=1:8,
     a222(i)=a22(i)*b1(i);
end
>> c2=ifft(a222,8);
>> disp(c2);
   3.0000    8.0000   14.0000   21.0000   28.0000   17.0000    9.0000    5.0000
>> a3=[1 2 3 4 5 0 0 0];
>> a33=fft(a3,8);
>> for i=1:8,
   a333(i)=a33(i)*b1(i);
end
>> c3=ifft(a333,8);
>> disp(c3);
   3.0000    8.0000   14.0000   21.0000   28.0000   17.0000    9.0000    5.0000
>> %disp(c4);
>> for i=1:5,
  d(i)=c1(i);
end
>> for i=6:8,
 d(i)=c2(i-5)+c1(i);
end
>> for i=9:10,
 d(i)=c2(i-5);
end
>> for i=11:13,
 d(i)=c3(i-10) +c2(i-5);
end
>> for i=14:18,
 d(i)=c3(i-10);
end
>> disp(d);
   3.0000    8.0000   14.0000   21.0000   28.0000   20.0000   17.0000   19.0000   21.0000   28.0000   20.0000
17.0000   19.0000   21.0000   28.0000   17.0000    9.0000    5.0000
```

The output sequence is $y_1(n) = [\ 3\ \ 8\ \ 14\ \ 21\ \ 28\ \ 17\ \ 9\ \ 5\ ]$

$y_2(n) = [\ 3\ \ 8\ \ 14\ \ 21\ \ 28\ \ 17\ \ 9\ \ 5\ ]$

$y_3(n) = [\ 3\ \ 8\ \ 14\ \ 21\ \ 28\ \ 17\ \ 9\ \ 5\ ]$

The resultant output using MATLAB program is
[3 8 14 21 28 20 17 19 21 28 20 17 19 21 28 17 9 5].
This result is same as obtained after adding the overlapping segment values as given below.

Output of length L + M – 1 = 8 for

Block 1: Block 2: Block 3:

```
3  8  14  21  28  17  9  5

            +    3  8  14  21  28   17  9  5

                               +    3  8  14  21  28  17  9  5
```

Add overlapped output

3 8 14 21 28 20 17 19 21 28 20 17 19 21 28 17 19 5

4. Overlap-Save Algorithm

The algorithm for overlap-save method can be written as follows:

Step 1: Take L samples of the data sequence $x(n)$.

Step 2: Append M – 1 last sample from the first block at the start of the second block and so on so that the length of each block is L + M – 1.

Step 3: For the first block, there is no previous block, append zeros at the start of this block.

Step 4: Append L – 1 extra zeros to the FIR of the filter so that its length is L + M – 1.

Step 5: Convolve the two sequences circularly using FFT to obtain the output.

Step 6: Repeat steps 1 to 5 for all the blocks.

Step 7: Accumulate the result by appending samples of the convolved output for successive blocks. Discard first M – 1 samples from result of every block and then append M – 1 samples of the first block at the end of the accumulated result by circular shifting as shown in Fig.3

Step 8: Overlap the input data samples and save the rest of the non-overlapping result samples.

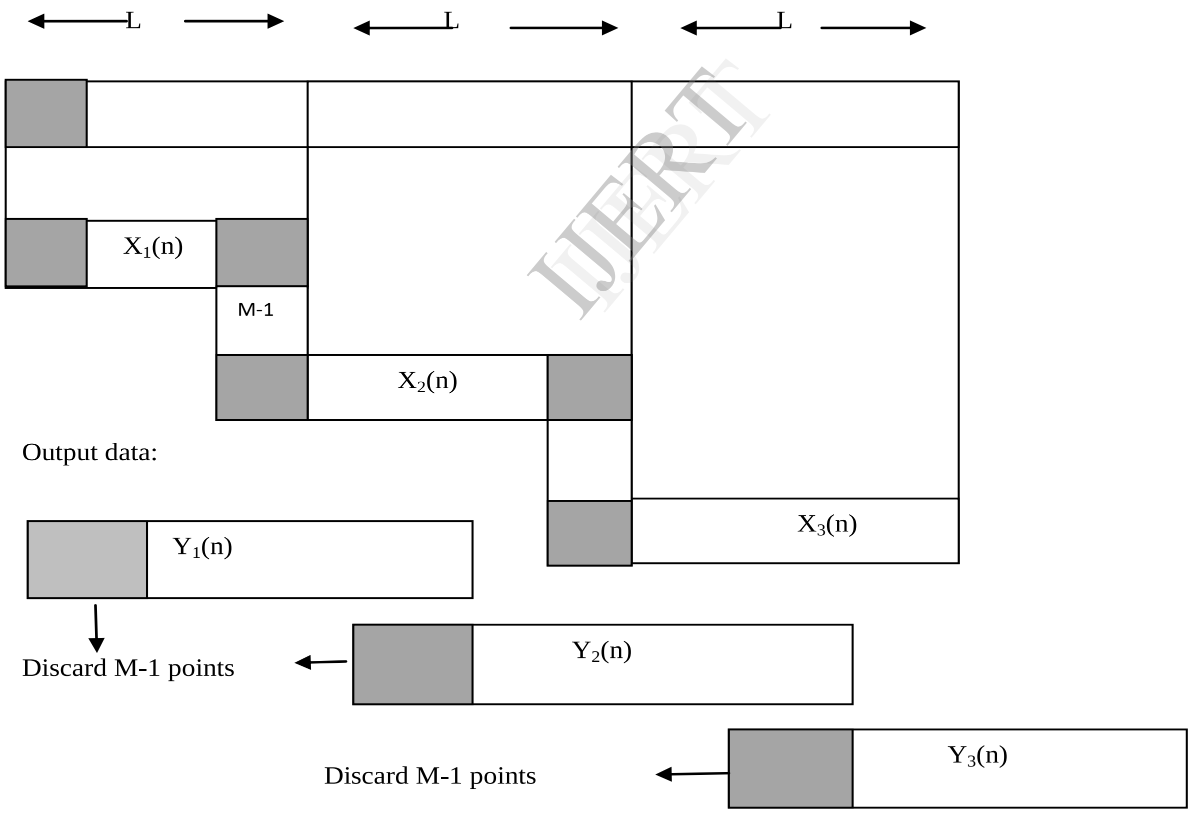

Fig.3 Linear FIR filtering by the overlap-save method.

For example, let the input sequence be $x(n)$ = [1 2 3 4 5 1 2 3 4 5 1 2 3 4 5].Let the FIR filter sequence be $h(n)$ = [3 2 1 1]. Let us calculate the convolution using overlap-save method. We will use block size for the data as L = 5 and M = 4. We have to append three zeros to the data block and 4 zeros to $h(n)$. Therefore, the length of both the sequences is L + M – 1 = 5 + 4 – 1 = 8. When we use FFT, N is selected as some exponential of 2. The data blocks will be

$$x_1(n) = [0\ \ 0\ \ 0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5]$$
$$x_2(n) = [3\ \ 4\ \ 5\ \ 1\ \ \ 2\ \ 3\ \ \ 4\ \ 5\,]$$
$$x_3(n) = [\,3\ \ 4\ \ 5\ \ 1\ \ \ 2\ \ 3\ \ \ 4\ \ 5\,]$$
$$h(n) = [\,3\ \ 2\ \ 1\ \ 1\ \ \ 0\ \ 0\ \ 0\ 0\,]$$

5. MATLAB Program for Overlap-Save Algorithm

The result of convolution of data blocks is obtained by using a MATLAB program. A MATLAB program and the output sequence is as follows:

```
>> %overlap save algorithm
>> clear all;
>> a=[1 2 3 4 5 1 2 3 4 5 1 2 3 4 5];
>> b=[3 2 1 1 0 0 0 0];
>> c=conv(a,b);
>> disp(c);
    3    8   14   21   28   20   17   19   21   28   20   17   19   21   28   17    9    5    0    0    0    0
>> b1=fft(b,8);
>> a1=[0 0 0 1 2 3 4 5];
>> a11=fft(a1,8);
>> for i=1:8,
  a111(i)=a11(i)*b1(i);
end
>> c1=ifft(a111,8);
>> disp(c1);
  17.0000    9.0000    5.0000    3.0000    8.0000   14.0000   21.0000   28.0000
>> a2=[3 4 5 1 2 3 4 5];
>> a22=fft(a2,8);
>> for i=1:8,
  a222(i)=a22(i)*b1(i);
end
>> c2=ifft(a222,8);
>> disp(c2);
   26    27    31    20    17    19    21    28
>> a3=[3 4 5 1 2 3 4 5];
>> a33=fft(a3,8);
>> for i=1:8,
  a333(i)=a33(i)*b1(i);
end
>> c3=ifft(a333,8);
>> disp(c3);
   26    27    31    20    17    19    21    28
>> a4=[3 4 5 0 0 0 0 0];
>> a44=fft(a4,8);
>> for i=1:8,
  a444(i)=a44(i)*b1(i);
end
>> c4=ifft(a444,8);
>> disp(c4);
   9.0000   18.0000   26.0000   17.0000    9.0000    5.0000         0    0.0000
>> for i=1:5,
  d(i)=c1(3+i);
end
>> for i=6:10,
  d(i)=c2(i-2);
end
>> for i=11:15,
  d(i)=c3(i-7);
end
```

```
>> for i=16:20,
   d(i)=c4(i-12);
end
>> disp(d);
  Columns 1 through 19
    3.0000    8.0000   14.0000   21.0000   28.0000   20.0000   17.0000   19.0000   21.0000   28.0000   20.0000
17.0000   19.0000   21.0000   28.0000   17.0000    9.0000    5.0000         0
  Column 20
  0.0000
```

The result of MATLAB program is
[3 8 14 21 28 20 17 19 21 28 20 17 19 21 28 17 9 5].
The output convolved sequences obtained are $y_1(n) = [\,17\ \ 9\ \ 5\ \ 3\ \ 8\ \ 14\ \ 21\ \ 28\,]$
$y_2(n) = [\,26\ \ 27\ \ 31\ \ 20\ \ 17\ \ 19\ \ 21\ \ 28\,]$
$y_3(n) = [\,26\ \ 27\ \ 31\ \ 20\ \ 17\ \ 19\ \ 21\ \ 28\,]$

Let us now discard the first three output samples from each output sequence and save the non-overlapping outputs as shown in Table 1. First M – 1 samples of the first block are appended at the end by circular shifting. When we append M – 1 extra zeros at the beginning of the data block, the resulting output convolved sequence will also be circularly rotated by M – 1 samples. The block-wise output is as shown in Table 1. Hence the resultant output is

[3 8 14 21 28 20 17 19 21 28 20 17 19 21 28 17 9 5].

Table 1: Output of length L + M – 1 = 8 for three blocks.

Block: 1 $y_1(n)$	Block: 2 $y_2(n)$	Block: 3 $y_3(n)$
0 0 0 3 8 14 21 28 17 9 5 + 17 9 5	9 18 26 20 17 19 21 28 17 9 5 +17 9 5	9 18 26 20 17 19 21 28 17 9 5 +17 9 5
17 9 5 3 8 14 21 28	26 27 31 20 17 19 21 28	26 27 31 20 17 19 21 28
3 8 14 21 28	**20 17 19 21 28**	**20 17 19 21 28 17 9 5**

6. MATLAB Program for the plot of input and output signal

```
>> clear all;
>> x=input('enter the sequence x(n)=');
enter the sequence x(n)=[1 2 3 4 5 1 2 3 4 5 1 2 3 4 5]
>> h=input('enter the sequence h(n)=');
enter the sequence h(n)=[3 2 1 1]
>> y=conv(x,h);
>> figure; subplot(3,1,1);
>> stem(x); ylabel('Amplitude -->');
>> xlabel('(a)   n-->');
>> subplot(3,1,2);
>> stem(h); ylabel('Amplitude -->');
>> xlabel('(b)   n-->');
>> subplot(3,1,3);
>> stem(y); ylabel('Amplitude -->');
>> xlabel('(c)   n-->');
>> disp('The resultant signal is');y
The resultant signal is
y = 3     8    14    21    28    20    17    19    21    28    20    17    19    21    28    17     9     5
```

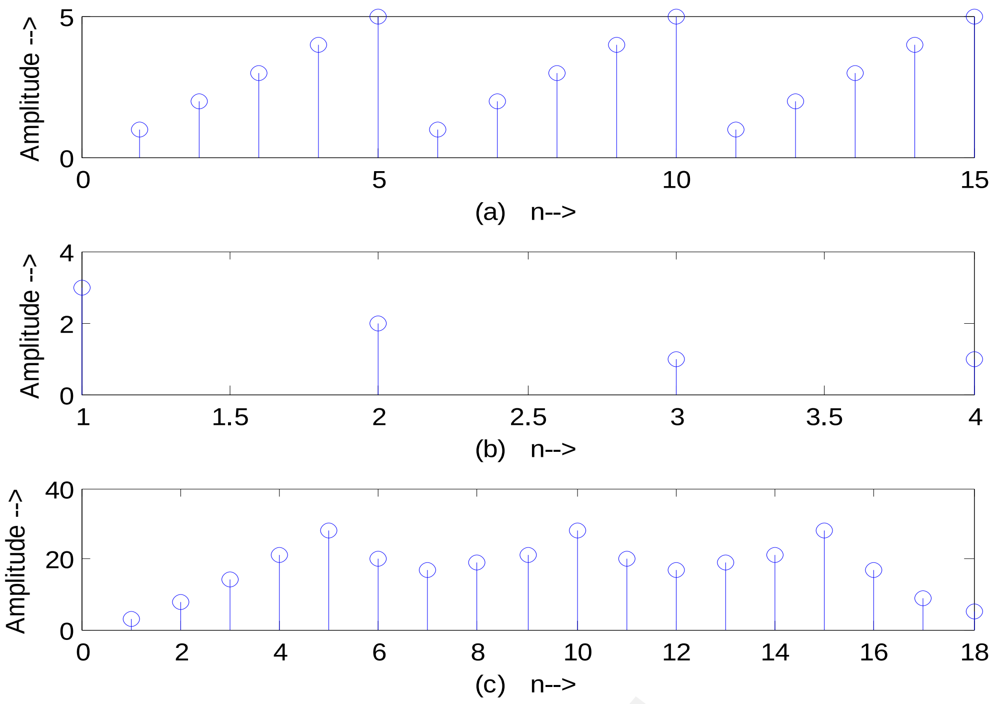

Fig. 4 (a) Plot of the input signal $x(n)$; (b) Plot of the input signal $h(n)$;
(c) Plot of the output signal $y(n)$.

7. Conclusion

We observe that the resultant output signal for overlap-add and overlap-save algorithm is the same. Hence, we can say that both the methods are equivalent. Combining the overlap-add and overlap-save methods with the use of FFT yields a very efficient algorithm for calculating convolution that is faster than direct calculation for lengths above 20 to 50. We have implemented the MATLAB Program to obtain the result for linear filtering operation in the frequency domain. The two signal segmentation methods, namely, the overlap-add and overlap-save methods are used to perform fast convolution by sectioning the long input sequence into blocks. The final convolution output result of the sequence is obtained by combining the partial convolution results generated from each block using MATLAB program. We obtained the result of the graph as shown in Fig. 4(a), (b) and (c) of plot of input and output signal using MATLAB program.

References

[1] James W.Cooley and James W.Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series”, Math. Comp.,Vol.19., April 1965, pp. 297-301.

[2] Leo I.Bluestein, “A Linear Filtering Approach to the Computation of Discrete Fourier Transform”, IEEE Transactions of Audio and Electroacoustics, Vol.AU-18, No.14, December 1970, pp. 451-455.

[3] M.Vetterli, “Running FIR and IIR Filtering using Multirate Filter Banks”, IEEE Trans. Acoust. Speech Signal Process., Vol. ASSP-36, No.5, May 1988, pp. 730-738.

[4] James W.Cooley, Peter A.W.Lewis and Peter D.Welch, “ The Fast Fourier Transform and its Applications”, IEEE Transactions on Education, Vol., 2 No.1, March 1969, pp. 27-34.

[5] P.Duhamel and M.Vetterli, “Fast Fourier Transforms: A Tutorial View and a State of Art”, Signal Processing, Elsevier Science publishers, 19(1990), pp. 259-299.

[6] S.Winogard, “On Computing the Discrete Fourier Transform”, Proc. Nat. Acad. Sci., U.S.A, Vol.73, April 1976, pp. 1005-1006.

[7] Z.J.Mou and P.Duhamel, “Fast FIR filtering: Algorithm and Implementation”, Signal Processing, Vol.13, No.4, December 1987, pp. 377-384.

[8] Harris, F.J., “Time domain signal processing with the Discrete Fourier Transform, in D.F.Elliot edition., Handbook of Digital Signal Processing”, Ch.8, Academic Press, New York,1987, pp. 633-699.

[9] Stockham,T.G., “High Speed Convolution and Correlation in AFIPS”, Conf. Proc., Spring Joint Computer Conference, 1966. Vol. 28, pp. 229-233.

[10] Nussbaumer, H.J., “Fast Fourier Transform and Convolution Algorithms”, Springer-Verlag, New York, 1982.

[11] Helms, H.D., “Fast Fourier Transform method of computing Difference Equations and simulating filters”, IEEE Trans. Audio Electroacoust., AU-15, June 1967, pp. 85-90

[12] Blahut, R.E., “Fast Algorithms for Digital Signal Processing”, Addison-Wesley, Reading, MA, 1985.

[13] White, S.A., “Applications of Distributed Arithmetic to Digital Signal Processing”, IEEE ASSP Mag., 6(3), July 1989, pp. 4-19.

[14] Burrus, C.S., “Digital Filter Structures described by distributed arithmetic”, IEEE Trans. Circuits Syst., CAS-24(12), December 1977, pp. 674-680.

[15] Robiner, L.R and Gold, B., “Theory and Application of Digital Signal Processing”, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[16] Myers, D.G., “Digital Signal Processing: Efficient Convolution and Fourier Transform Techniques”, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[17] John G. Proakis and Dimitris G. Manolakis, “Digital Signal Processing: Principles, Algorithms and Applications”, Fourth Edition, Pearson Education, Inc., 2007.

[18] Ivan W.Selesnick and Burrus, C.S., “Fast Convolution and Filtering”, CRC Press LLC, 1999.

[19] Shaila D.Apte, “Digital Signal Processing”, Second Edition, Wiley India, August, 2009.

[20] Salivahanan, S and Ganapriya, C., “Digital Signal Processing”, Second Edition, Tata McGraw Hill, 2011.