Calculating the DTFS of signals

Fourier Series

- Periodic signal with period T or N
- Synthesis equation:

$$x(t) = \sum_{k=-\infty}^{+\infty} a_k e^{jk\left(\frac{2\pi}{T}\right)t} \quad v.s. \quad x[n] = \sum_{k=0}^{N-1} a_k e^{jk\left(\frac{2\pi}{N}\right)n}$$

- Analysis equation:

$$a_k = \frac{1}{T}\int_T x(t)e^{-jk\left(\frac{2\pi}{T}\right)t}dt \quad v.s. \quad a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\left(\frac{2\pi}{N}\right)n}$$

Why not $-\infty \sim +\infty$?

Summation of N harmonic components

$$\boxed{a_k \xleftrightarrow{relation?} a_{k+N}}$$

Complexity Analysis

- Suppose we know the matrix

$$E(n,k) = e^{-jk\left(\frac{2\pi}{N}\right)n}$$

- How many multiplications & additions are needed to calculate Fourier series $[a_0, a_1, \dots, a_{N-1}]$

计算次数

One Period

$$a_0 = \frac{1}{N}\sum_{n=0}^{N-1} x[n]E(n,0)$$

$$\dots$$

$$a_{N-1} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]E(n,N-1)$$

N+1个乘法

Each: $N+1\ \times$; $N-1\ +$

Total: $N(N+1)\ \times$; $N(N-1)\ +$

Complexity: $O(N^2)$

Fast Fourier Transform (FFT)

- Fast Fourier Transform: Calculation of Fourier series (transform) can be speeded up
  - Complexity reduces to $O(NlogN)$

$$E(n,k) = e^{-jk\left(\frac{2\pi}{N}\right)n}$$

$N = 4$

$a_0 = (x[0]E(0,0) + x[1]E(1,0) + x[2]E(2,0) + x[3]E(3,0))/N$

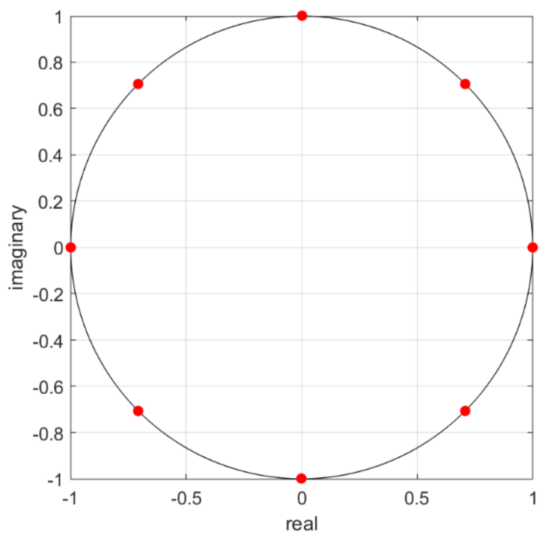
$a_2 = (x[0]E(0,2) + x[1]E(1,2) + x[2]E(2,2) + x[3]E(3,2))/N$

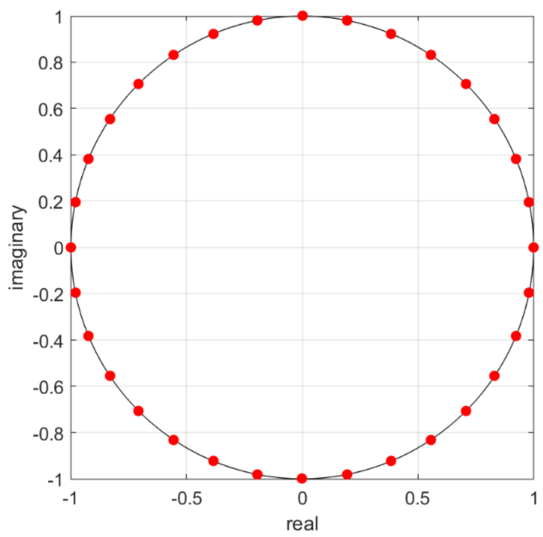
$a_1 = (x[0]E(0,1) + x[1]E(1,1) + x[2]E(2,1) + x[3]E(3,1))/N$

$a_3 = (x[0]E(0,3) + x[1]E(1,3) + x[2]E(2,3) + x[3]E(3,3))/N$

$E(1,1) = -E(1,3); E(2,1) = E(2,3); E(3,1) = -E(3,3);$

Reference: http://en.wikipedia.org/wiki/Cooley–Tukey_FFT_algorithm


imaginary
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
real


imaginary
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
real

- To calculate the DTFS and FFT of a 1024*1024 image:

CPU	Clock Frequency	DTFS	FFT
1941	60 Hz	152.3 y	271.4 d
1971 (4004)	108KHz	30.8 d	3.6 h
1978 (8086)	10MHz	8.0 h	2.3 min
1982 (80286)	20MHz	4.0 h	1.2min
1985 (80386)	33MHz	2.4h	42.6s
1989 (80486)	100MHz	48.0min	14.1s
1995 (Pentium)	200MHz	24.0min	7.0s
1999 (Pentium Ⅲ)	450MHz	10.7min	3.1s
2000 (Pentium 4)	1.4GHz	3.4min	1.0s
2001 (Pentium 4)	2GHz	2.4min	0.7s

Matlab Function: fft()

- fft(): compute DTFS coefficients from signals

```
>> help fft
                    N
     X(k) =       sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
                   n=1
```

$$a_k = \sum_{n=1}^{N} x[n]e^{-j(k-1)\left(\frac{2\pi}{N}\right)(n-1)}, 1 \le k \le N$$

- Compare with our definition: $a_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\left(\frac{2\pi}{N}\right)n}, 0 \le k \le N-1$
- Calculate the DTFS of vector x: `a = (1/N) * fft(x)`

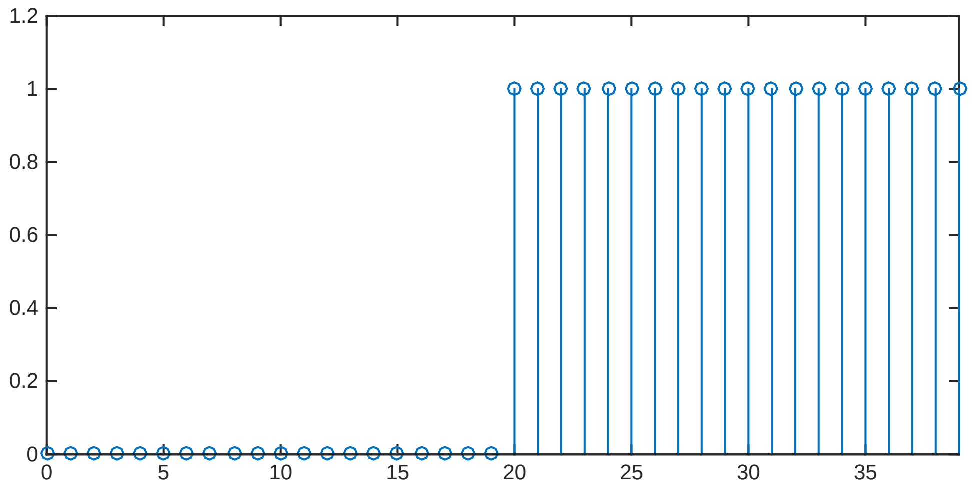
Matlab Function: ifft()

- ifft(): reconstruct signals from DTFS coefficients
- Matlab definition: $$x[n] = \frac{1}{N}\sum_{k=1}^{N} a_k e^{j(k-1)\left(\frac{2\pi}{N}\right)(n-1)}$$
- Compare with our definition: $$x[n] = \sum_{k=0}^{N-1} a_k e^{jk\left(\frac{2\pi}{N}\right)n}$$
- Calculate the DTFS of vector x:

```
x = N * ifft(a)
```

Example

- Periodic DT rectangular wave with period = 40



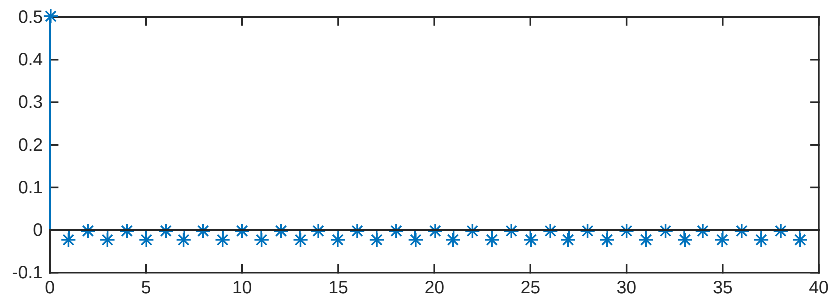

```
x=[zeros(1,20) ones(1,20)];
stem(0:39, x);
xlim([0 39]);
ylim([0 1.2]);
```

```
A = fft(x) / length(x);
subplot(2,1,1), stem(0: length(x)-1,real(A),'*-');
subplot(2,1,2), stem(0: length(x)-1,imag(A),'*-');
```

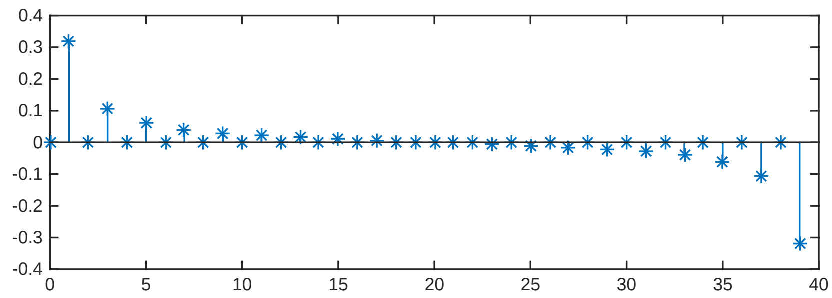
Plot the real part

Plot the imaginary part

Real Part:



Imaginary Part:

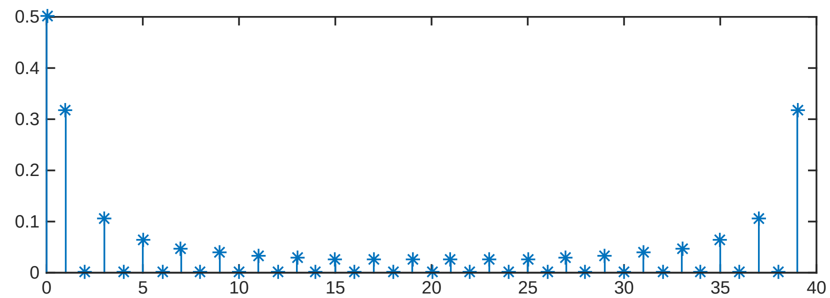


```
subplot(2,1,1), stem(0: length(x)-1,abs(A),'*-');
subplot(2,1,2), stem(0: length(x)-1,angle(A),'*-');
```

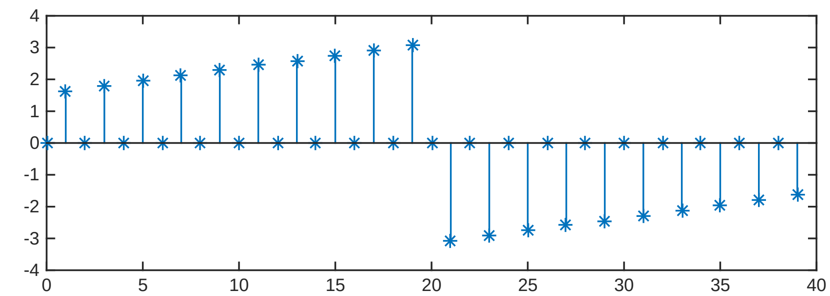
Plot the magnitude

Plot the phase

Magnitude:



Phase:

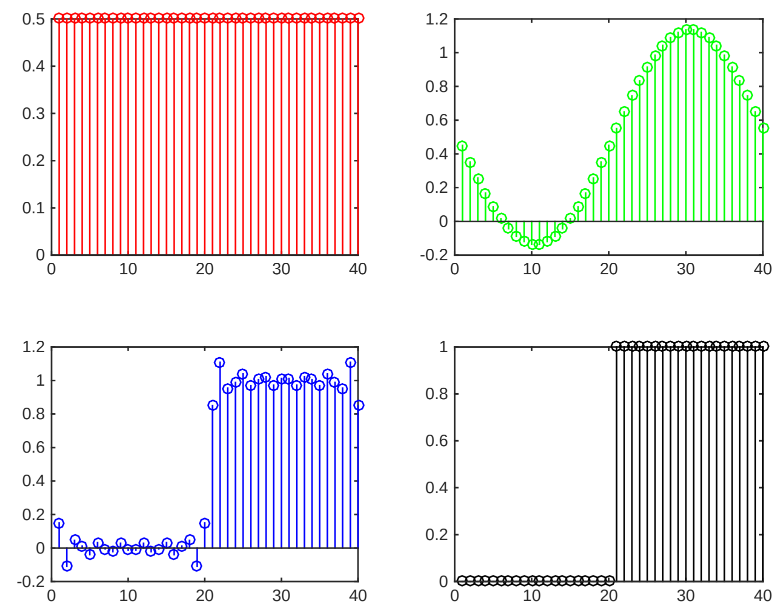


Matlab definition: $a_k = \sum_{n=1}^{N} x[n] e^{-j(k-1)\left(\frac{2\pi}{N}\right)(n-1)}$, $1 \le k \le N$

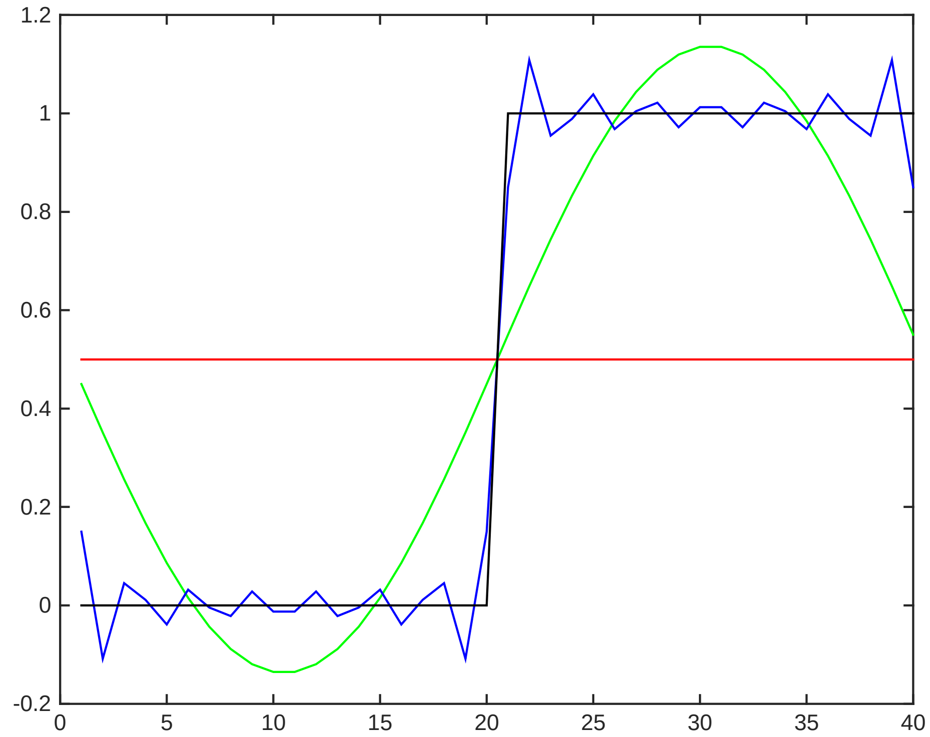
textbook definition: $a_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk\left(\frac{2\pi}{N}\right)n}$, $0 \le k \le N-1$

```
newA1 = [A(1) zeros(1,39)];
newA2 = [A(1) A(2) zeros(1,37) A(40)];
newA3 = [A(1) A(2:15) zeros(1,11) A(27:40)];


subplot(2,2,1), stem(1:40,ifft(newA1)*40, 'r');
subplot(2,2,2), stem(1:40,ifft(newA2)*40, 'g');
subplot(2,2,3), stem(1:40,ifft(newA3)*40, 'b');
subplot(2,2,4), stem(1:40,x, 'k');
```



```
plot(1:40,ifft(A1)*40, 'r', 1:40,ifft(A2)*40, 'g', 1:40,ifft(A3)*40, 'b', 1:40,x, 'k');
```



Lab Assignment 3 (b)

- Read tutorial 3.1, 3.2 & 3.3 by yourself
- 3.5, 3.8(c-f) & 3.10

Lab Assignment 3 summary

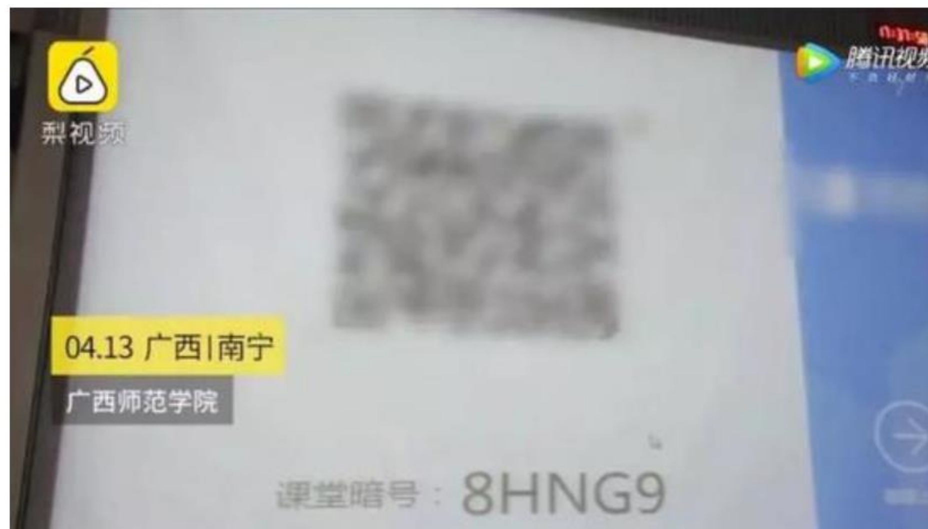
- Read tutorial 3.1, 3.2 & 3.3 by yourself
- 3.5, 3.8, 3.9 & 3.10
- Submit your report + codes onto Blackboard before 10:00 am Nov. 4th

新浪教育 新浪教育 > 高考 > 正文

逃课大学生的末日！老师启用“点名神器”

2018年04月19日 10:19 中青在线

一位女生说，第一次用这种方式点名就被点到了，转发抽奖都没中过，内心很崩溃。



Hints & Tips

- 3.5
  - The DTFS coefficients of a periodic discrete-time signal with period N = 5:

$$a_0 = 1,\ a_2 = a_{-2}^* = e^{j\pi/4},\ a_4 = a_{-4}^* = 2e^{j\pi/3}.$$

  - a1 = ?
  - a3 = ?

3.5 Advanced Problem

Advanced Problem

For this problem, you will write a function which computes the DTFS coefficients of a periodic signal. Your function should take as arguments the vector **x**, which specifies the values of the signal $x[n]$ over one period, and **n_init** which specifies the time index n of the first sample of **x**. Your function should return the vector **a** containing the DTFS coefficients a_0 through a_{N-1}, where N is the number of samples in **x**, or equivalently, the period of $x[n]$. The first line of your M-file should read

```
function a = dtfs(x,n_init)
```

```
>> dtfs([1 2 3 4],0)
ans =
   2.5000          -0.50
>> dtfs([1 2 3 4],1)
ans =
   2.5000           0.50
>> dtfs([1 2 3 4],-1)
ans =
   2.5000          -0.50
>> dtfs([2 3 4 1],0)
ans =
   2.5000          -0.50
```

[1,2,3,4]
[4,1,2,3]
[2,3,4,1]
[2,3,4,1]

abs(dtfs(*,*)) difference?
angle(dtfs(*,*)) difference ?

- 3.10 (b) & (c)
  - Suggested in the lab book: measure the number of operations by using the internal flops 'flops'
  - To measure the elapsed time by a function, you can choose one of the followings:
    - etime
    - tic, toc
    - timeit

推荐这两种

```
x = (0.9).^[0:N-1];
flops(0);
X = dtfs(x,0);
c = flops;
```

flops：This is an obsolete function

- Measure the function performance difference by comparing the time cost by functions rather than numbers of floating point operations.
- Try to keep the PC conditions unchanged while measuring the functions timecosts

- etime

```
t0 = clock;
X = dtfs(x,0);
dtfstime = etime(clock,t0);

t1 = clock;
X2 = fft(x);
ffttime = etime(clock,t1);
```

The clock function returns the current date and time as a date vector, system clock

The command etime will allow you to measure the elapsed time between the start and finish of your implementation

etime(T1,T0) returns the time in seconds that has elapsed between vectors T1 and T0

- tic, toc

```
tic
X = dtfs(x,0);
toc

tic
X2 = fft(x);
toc
```

tic starts a stopwatch timer to measure performance. The function records the internal time at execution of the tic command. Display the elapsed time with the toc function

- timeit

```
f1 = @()dtfs(x,0);
t1_timeit = timeit(f1)

f2 = @()fft(x)
t2_timeit = timeit(f2)
```

t = timeit(f) measures the typical time (in seconds) required to run the function specified by the function handle f

To measure performance, it is recommended to use the timeit or tic and toc functions

If your PC is so good that the time cost by fft is always 0,
try to increase the value of N. ☺
Or, put your codes in a for-loop*, … …*

Tips

- 3.10 Periodic Convolution with the FFT
  - Periodic convolution in time domain is equivalent to multiplication in frequency domain

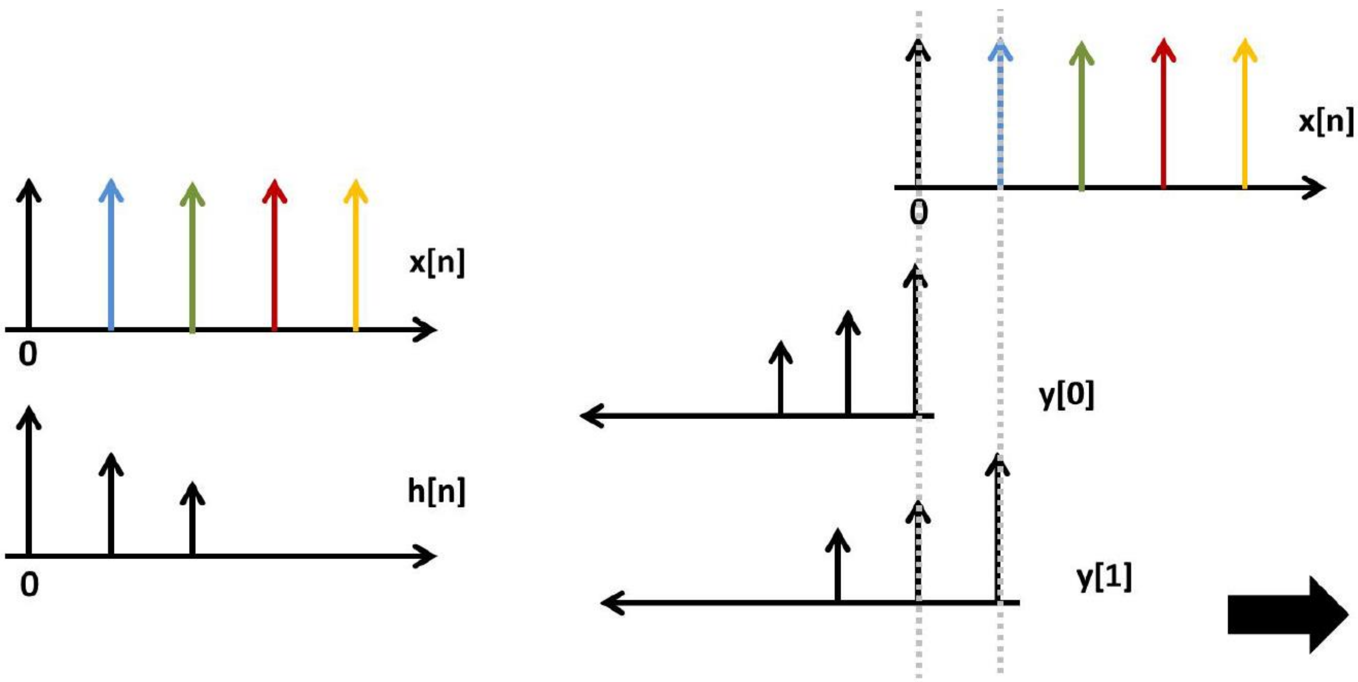
$$x[n] \otimes \hat{h}[n] = \sum_{r=0}^{N-1} x[r]\hat{h}[n-r] \Leftrightarrow Na_k h_k$$

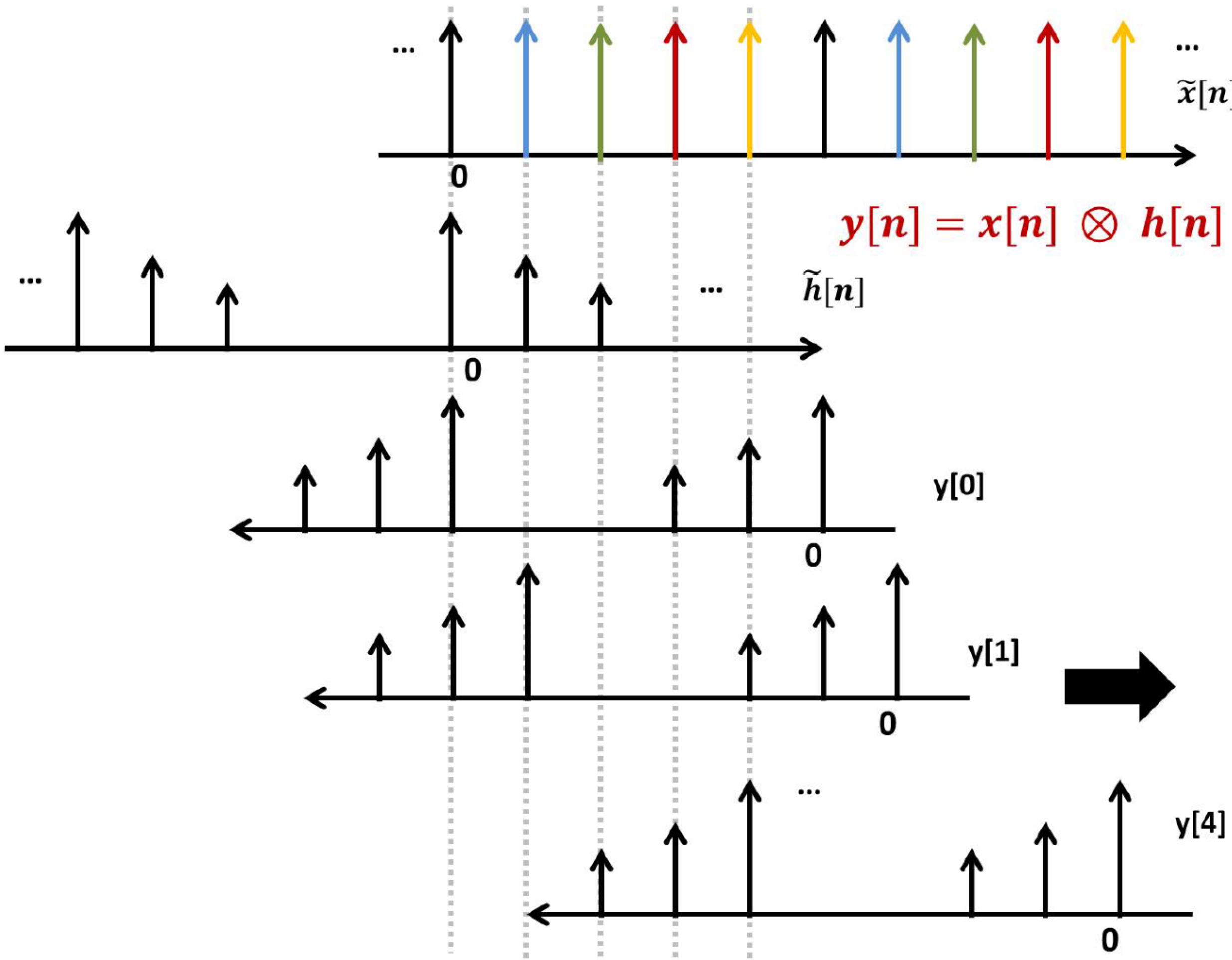
Table 3.2 in Textbook
Section 3.7.1 multiplication (property)

$$y[n] = x[n] * h[n] = x[n] \otimes \hat{h}[n]$$

$\hat{h}[n]$ **is a periodic version of h[n]**

`conv([x x],h)`. The periodic convolution can be extracted from a portion of this signal.


x[n]
0
h[n]
0
x[n]
0
y[0]
y[1]


...
...
$\widetilde{x}[n]$
0
$y[n] = x[n] \otimes h[n]$
...
...
$\widetilde{h}[n]$
0
y[0]
0
y[1]
0
...
y[4]
0

- Any question?