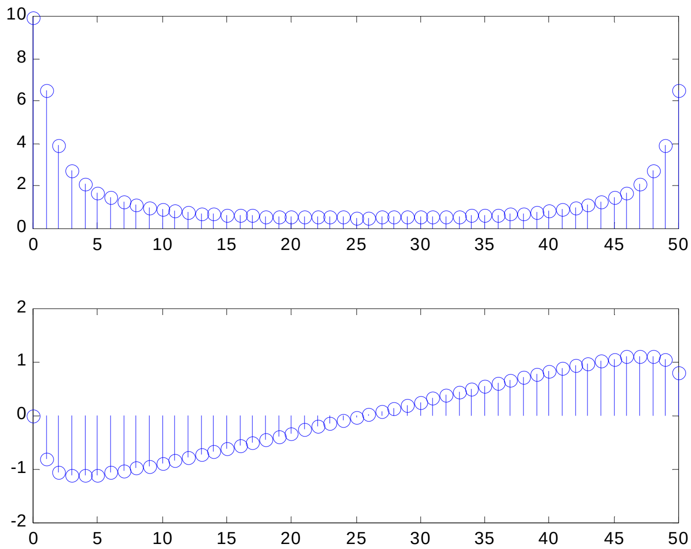
Chapter 5　DFS

南台科大　　趙春棠

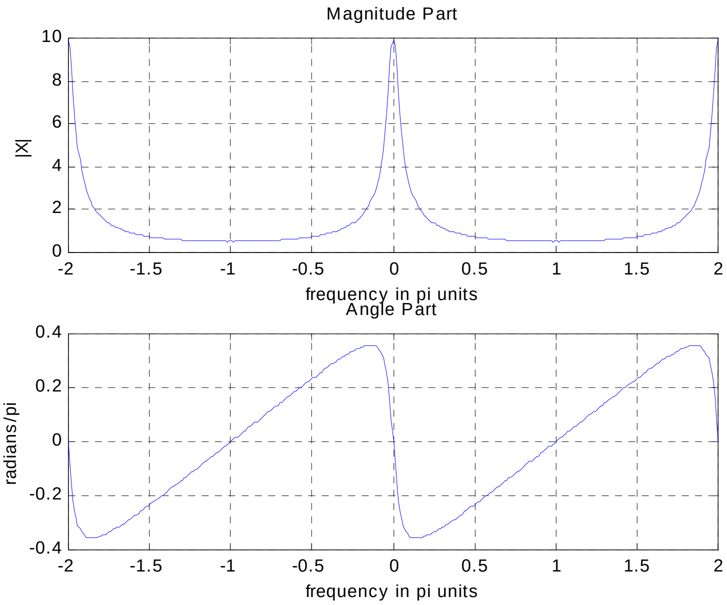
function [Xk] = dfs(xn,N)

```
% Computes Discrete Fourier Series Coefficients
% Xk = DFS coeff. array over 0 <= k <= N-1
% xn = One period of periodic signal over 0 <= n <= N-1
n =[0:1:N-1];  k= [0:1:N-1];
WN = exp(-j*2*pi/N);              % Wn factor
nk = n'*k;                               % creates a N by N matrix of nk values
WNnk = WN .^ nk;                  % DFS matrix
Xk = xn * WNnk;                   % row vector for DFS coefficients
```

Matlab: n=0:50; x=(0.9).^n; [Xk] = dfs(x,51); magXk=abs(Xk);　angXk=angle(Xk);
subplot(2,1,1); stem(n,magXk); subplot(2,1,2); stem(n,angXk);



比較： **Matlab:** n=0:50;　x=(0.9).^n;　[X]=plot_dtft(x,n);


Magnitude Part
|X|
10
8
6
4
2
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
frequency in pi units
Angle Part
radians/pi
0.4
0.2
0
-0.2
-0.4
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
frequency in pi units

頻率取樣

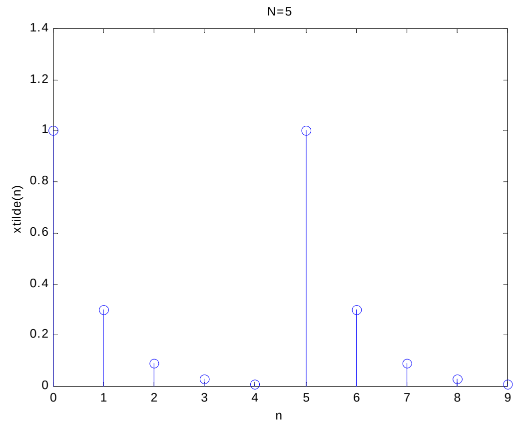
例： $x(n)=(0.3)^n u(n)$; $X(z)=\dfrac{1}{1-0.3z^{-1}}$

x=[1.0000 0.3000 0.0900 0.0270 0.0081 0.0024 0.0007
0.0002 0.0001 0.0000 0.0000 ….]

＊ 在 **X(z)**單位圓上取 **5** 點，並作 **IDFT**

Matlab: b=[1]; a=[1 –0.3]; N=5; k=0:1:N-1; w=2*pi*k/N;

```
Xk=freqz(b,a,w); xn=real(idfs(Xk,N));      %本例中，經 idfs(Xk,N)為實數
xtilde = xn'*ones(1,2); xtilde = (xtilde(:))';    % Periodic sequence
pn=0:9; stem(pn,xtilde);
xlabel('n'); ylabel('xtilde(n)'); title('N=5')
```



＊ 分析：

以上程式輸出 $\tilde{x}(n)$ =[1.0024 0.3007 0.0902 0.0271 0.0081]

事實上，$\tilde{x}(n)$ 與 x(n)的關係如下：

$$\tilde{x}(n)=\sum_{r=-\infty}^{\infty} x(n-rN)=\ldots+x(n+N)+x(n)+x(n-N)+\ldots$$

應證如下：

$\tilde{x}(0)$ = x(0)+x(5)+x(10)+…= 1.0000+0.0024+0.0000+….= 1.0024

$\tilde{x}(1)$ = x(1)+x(6)+x(11)+…= 0.3000+0.0007+0.0000+….= 0.3007

$\tilde{x}(2)$ = x(2)+x(7)+x(12)+…= 0.0900+0.0002+0.0000+….= 0.0902

$\tilde{x}(3)$ = x(3)+x(8)+x(13)+…= 0.0270+0.0001+0.0000+….= 0.0271

$\tilde{x}(4)$ = x(4)+x(9)+x(14)+…= 0.0081+0.0000+0.0000+….= 0.0081

註：1. 以上的例子是以一個無限長度的 x(n)，將其作 X(z)為例。若 x(n)為有限長度，以上公式亦成立

2. 觀察 $\tilde{x}(n)$ 與 x(n)之間的誤差，提供給我們今後 x(n)該取幾點做參考。似乎直接看 x(n)的訊號大小，也大概可看出。

DFT 與 DTFT 及 DFS

＊ **DFT** 其實與 **DFS** 的理論是一模一樣的，只不過原本週期性（週期 **N**）的離散訊號（**$x((n))_N$**），如今只需簡化討論 **0<= n <=(N-1)** 的範圍內（**x(n)**）即可，即所謂的 **N-point sequence**，進一步連 **DFT X(k)** 亦然。

＊ **$x((n))_N$ = x(n mod N)**

function m =mod(n,N)

```
% Compute m = (n mod N) index
m =rem(n,N);
m = m+N;
m = rem(m,N);
```

說明： 目的在使 $\tilde{x}(n)$ =$x((n))_N$ 能以 x(n)表示即可

例：

rem(12,7)

5

rem(-12,7)

-5

mod(12,7)

5

mod(-12,7)

2

利用 mod 可得知若一個週期爲 7 的序列 $\tilde{x}(n)$ ，則 $\tilde{x}(12)$ =x(5)，而 $\tilde{x}(-12)$ =x(2)

＊ 在 **0<= n <=(N-1)** 的範圍內，**DFS** 與 **DFT** 是相同的

function[Xk]=df**t**(xn,N) 與 function[Xk]=df**s**(xn,N) 程式內容完全相同

function[Xk]=**i**df**t**(xn,N) 與 function[Xk]=**i**df**s**(xn,N) 程式內容完全相同

＊ **Zero-padding operation**

以 x1(n)=[1 1 1 1] 以及 x2(n)=[1 1 1 1 **0 0 0 0**]爲例

此時若將 x1(n)及 x2(n)作 **DTFT** ，所得的 **$X(e^{jw})$** **是相同**的哦！

而在前述 DFS 的研討中，可推得知若將 x1(n)作 DFT，可得四點的 $X_4(k)$，此四點即相當於在 $X(e^{jw})$ 的 **0~2*pi 間取四點**

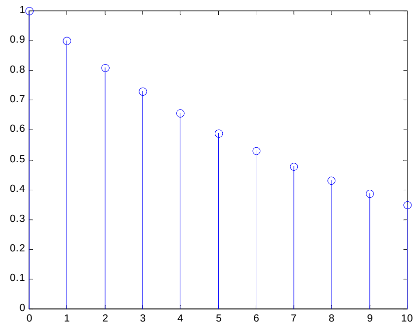
同理，若將 x2(n)作 DFT，將可得八點的 $X_8(k)$，此八點即相當於在 $X(e^{jw})$ 的 **0~2*pi 間取八點**

如此看來，如果想利用 DFT 取代 DTFT，而得到 0~2*pi 間適當的取樣數，以增加精確度，則當 x(n)序列點數太少時，可利用以上 zero-padding 方法，得到適當的結果

DFT　N-point Seauence

例： x[n]=[0.9^0　0.9^1　0.9^2　0.9^3　0.9^4 … 0.9^{10}];

技巧： 由於 DFT 的輸入 x(n)只考慮 0<= n <=(N-1) 的範圍內，但以下的推導，請利用「週期性」先將訊號擴展，經運算後，再取 0<= n <=(N-1) 的範圍

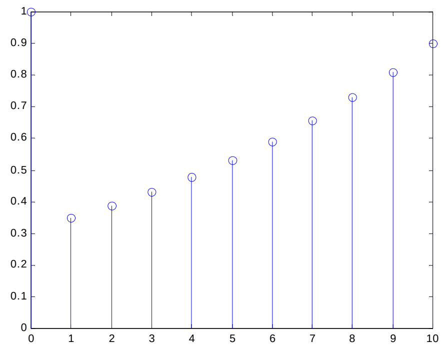


（Matlab: n=0:10; x=0.9.^n; stem(n,x);）

＊ **x((-n))$_N$**

先將以上 x(n)訊號週期性展開 => 取 x(-n) => 取 N 點（0<= n <=(N-1)）

Matlab: n=0:10; x=0.9.^n; y=x(mod(-n,11)+1); stem(n,y);



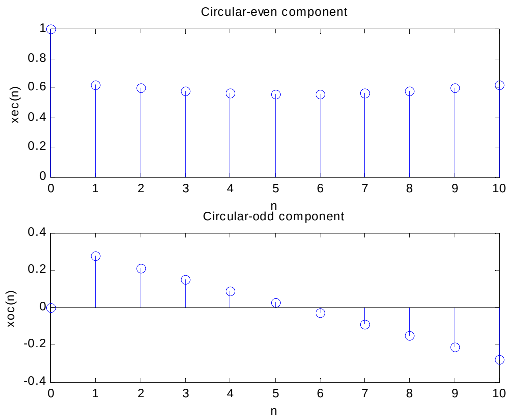
註：n =	0	1	2	3	4	5	6	7	8	9	10
-n =	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
mod(-n,11)=	0	10	9	8	7	6	5	4	3	2	1

註： 有關 DFT[x((-n))$_N$] 略

＊ **x(n)分解爲「循環-奇」與「循環-偶」兩部分**

```
function [xec, xoc] = circevod(x)
% signal decomposition into circular-even and circular-odd parts
if any(imag(x) ~= 0)
    error('x is not a real sequence')
end
N = length(x); n = 0:(N-1);
xec = 0.5*(x + x(mod(-n,N)+1));
xoc = 0.5*(x - x(mod(-n,N)+1));
```

Matlab: n=0:10; x=0.9.^n; [xec,xoc]=circevod(x);
subplot(2,1,1); stem(n,xec); title('Circular-even component'); xlabel('n'); ylabel('xec(n)');
xa=0.*n; hold on; plot(n,xa,'k'); hold off;
subplot(2,1,2); stem(n,xoc); title('Circular-odd component')
xlabel('n'); ylabel('xoc(n)'); hold on; plot(n,xa,'k'); hold off;



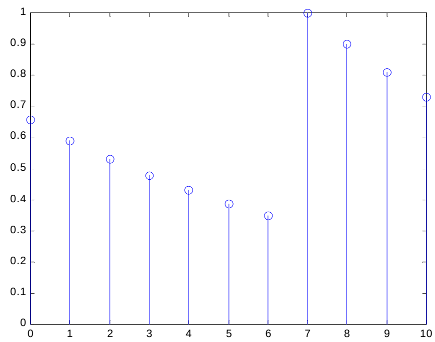
＊ Circular shift of a sequence

function y = cirshift(x,m,N)

```
if length(x) > N
    error('N must be >= the length of x')
end
x = [x zeros(1,N-length(x))];
n = [0:1:N-1];
n = mod(n-m,N);
y = x(n+1);
```

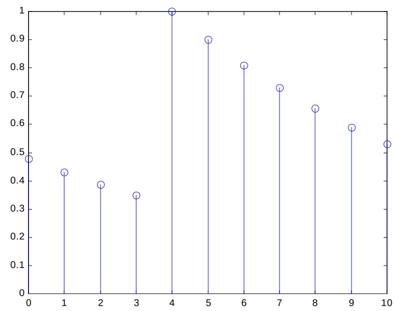
例： **x(n)** 向左循環移位 **4** 次得 $\mathbf{x((n+4))_{11}R_{11}(n)}$

Matlab: n=0:10; x=0.9.^n; y=cirshift(x,-4,11); stem(n,y);



例： **x(n)** 向右循環移位 **4** 次得 $\mathbf{x((n+4))_{11}R_{11}(n)}$

Matlab: n=0:10; x=0.9.^n; y=cirshift(x,4,11); stem(n,y);



Circular convolution

例： **x1=[1 3 2]; N1=3 點**

x2=[2 3 1 4 1]; N2=5 點　max(N1,N2)=5;　N1+N2-1=7

```
function y = circonvt(x1,x2,N)
% N-point circular convolution between x1 and x2: (time-doamin)
if length(x1) > N
    error('N must be >= the length of x1')
end
% Check for length of x2
if length(x2) > N
    error('N must be >= the length of x2')
end
x1=[x1 zeros(1,N-length(x1))];
x2=[x2 zeros(1,N-length(x2))];
m = [0:1:N-1];
x2 = x2(mod(-m,N)+1);
H = zeros(N,N);
for n = 1:1:N
    H(n,:) = cirshift(x2,n-1,N);
end
y = x1*H';
```

例： x1=[1 3 2]; x2=[2 3 1 4 1]; y = circonvt(x1,x2,4);

??? Error using ==> circonvt

N must be >= the length of x2　＊ **N >= max(N1,N2)** 必須成立

例 **A**： x1=[1 3 2]; x2=[2 3 1 4 1]; y = circonvt(x1,x2,5);

y = 13 11 14 13 15

例 **B**： x1=[1 3 2]; x2=[2 3 1 4 1]; y = circonvt(x1,x2,6);

y = 4 9 14 13 15 11

例 **C**： x1=[1 3 2]; x2=[2 3 1 4 1]; y = circonvt(x1,x2,7);

y = 2 9 14 13 15 11 2

例 **D**： x1=[1 3 2]; x2=[2 3 1 4 1]; y = circonvt(x1,x2,8);

y = 2 9 14 13 15 11 2 0

例 **E**： x1=[1 3 2]; x2=[2 3 1 4 1]; y = conv(x1,x2);

y = 2 9 14 13 15 11 2

例 **A** 的變形： x1=[1 3 2]; x2=[0 0 2 3 1]; y = circonvt(x1,x2,5);

y = 9 2 2 9 14

以上做法爲循環旋積的**時域**求解，以下示範**頻域**求解，
亦即分別對 x1(n)及 x2(n)作 **DFT**，然後將 X1(k)與 X2(k)相乘後，取 **IDFT**

例 **A**： x1=[1 3 2 0 0]; x2=[2 3 1 4 1]; X1=dft(x1,5); X2=dft(x2,5);
Y=X1.*X2; y =idft(Y,5);
結果：X1 = 6.0000 0.3090 - 4.0287i -0.8090 + 0.1388i -0.8090 - 0.1388i
0.3090 + 4.0287i
X2 = 11.0000 -0.8090 - 0.1388i 0.3090 - 4.0287i 0.3090 + 4.0287i
-0.8090 + 0.1388i
Y = 66.0000 -0.8090 + 3.2164i 0.3090 + 3.3022i 0.3090 - 3.3022i
-0.8090 - 3.2164i
y = 13.0000 + 0.0000i 11.0000 + 0.0000i 14.0000 - 0.0000i
13.0000 - 0.0000i 15.0000 - 0.0000i 與前述結果完全吻合
註：以上若 x1=[1 3 2]; 將有誤，因爲 dft(x,N)中，x 與 N 長度需相同

例 **B**： x1=[1 3 2 0 0 0]; x2=[2 3 1 4 1 0]; X1=dft(x1,6); X2=dft(x2,6);
Y=X1.*X2; y =idft(Y,6);
y = 4.0000 - 0.0000i 9.0000 - 0.0000i 14.0000 - 0.0000i 13.0000 + 0.0000i
15.0000 + 0.0000i 11.0000 + 0.0000i 與前述結果完全吻合

例 **C**： x1=[1 3 2 0 0 0 0]; x2=[2 3 1 4 1 0 0]; X1=dft(x1,7); X2=dft(x2,7);
Y=X1.*X2; y =idft(Y,7);
y = 2.0000 - 0.0000i 9.0000 - 0.0000i 14.0000 + 0.0000i 13.0000 + 0.0000i
15.0000 + 0.0000i 11.0000 + 0.0000i 2.0000 + 0.0000i 與前述結果完全吻合

例 **D**： x1=[1 3 2 0 0 0 0 0]; x2=[2 3 1 4 1 0 0 0]; X1=dft(x1,8); X2=dft(x2,8);
Y=X1.*X2; y =idft(Y,8);
y = 2.0000 - 0.0000i 9.0000 - 0.0000i 14.0000 - 0.0000i 13.0000 + 0.0000i
15.0000 + 0.0000i 11.0000 + 0.0000i 2.0000 + 0.0000i 0.0000 + 0.0000i
與前述結果完全吻合

結語： 循環旋積可以用 DFT 及 IDFT 來作，然而不管是時域求解或是頻域求解，取的點數 N 太小是不行的，當 **N=(N1+N2-1)**時沒有誤差

Block convolution

前提： x=[2 3 1 4 1 –1 2 3 1 2]; h=[1 3 2];

y=conv(x,h)

y = 2 9 14 13 15 10 1 7 14 11 8 4

y=circonvt(x,h,12) 結果與上述相同

將 **x[n]**分段： 每段取 **6** 點，每段輸出亦取六點

x1=[0 0 2 3 1 4]; h=[1 3 2]; y1=circonvt(x1,h,6)

y1 = 14 8 <u>2 9 14 13</u>

x2=[1 4 1 –1 2 3]; h=[1 3 2]; y2=circonvt(x2,h,6)

y2 = 14 13 <u>15 10 1 7</u>

x3=[2 3 1 2 0 0]; h=[1 3 2]; y3=circonvt(x3,h,6)

y3 = 2 9 <u>14 11 8 4</u>

分析： 以 x1（N1=6）及 h（N2=3）來說，N=6=max(N1,N2)

故此時 x1 與 h 作 6 點循環旋積所得的結果 y1，其中 y1 的前 M-1（=2）點是錯誤的，故將 y1 的前 2 點結果捨去，y2、y3 亦然。

註：**M=min(N1,N2)=3**

```
function [y] = ovrlpsav(x,h,N)
% Overlap-Save method of block convolution
Lenx = length(x); M = length(h);
M1 = M-1; L = N-M1;
h = [h zeros(1,N-M)];
%
x =[zeros(1,M1), x, zeros(1,N-1)];% preappend (M-1) zeros
K = floor((Lenx+M1-1)/(L));                % # of blocks
Y = zeros(K+1,N);
% convloution with succesive blocks
for k=0:K
    xk = x(k*L+1:k*L+N);
    Y(k+1,:) = circonvt(xk,h,N);
end
Y = Y(:,M:N)';                             % discard the first (M-1) samples
y = (Y(:))';
```

利用以上 **ovrlpsav** 的做法：

x=[2 3 1 4 1 –1 2 3 1 2]; h=[1 3 2]; y= ovrlpsav(x,h,6);

y = 2 9 14 13 15 10 1 7 14 11 8 4

FFT

＊ 使用 **X=fft(x.N)** 計算 **FFT**，使用 **x=ifft(X,N)** 計算 **IFFT**

＃ fft 是以機器語言寫成，並無 .m 檔，
若 x 長度小於 N 則 x 會自動補零，成為 N 點序列
若使用 X=fft(x)，省略 N，則以 x 長度作為 N

＃ fft 是根據 mixed-radix 演算法，
當 N 是 2 的次方時，採用快速基-2 FFT 演算法；
當 N 不是 2 的次方時，N 將被分解成質數因子並採用較慢速之混合-基演算法
當 N 是質數時，fft 回到原始 DFT 演算法

＊ **FFT** 的計算效率

```
Nmax=2048;
fft_time=zeros(1,Nmax);
for n=1:1:Nmax
   x=rand(1,n);
   t=clock; fft(x); fft_time(n)=etime(clock,t);
end
n=1:1:Nmax;
plot(n,fft_time,'.');
xlabel('N'); ylabel('Time in Sec');
title('FFT execution times');
```

以上程式執行有問題，總之 **FFT** 使 **O(N^2)**改進為 **O(NlogN)**

＊ 快速旋積

＃ Matlab 的 conv 是利用 filter 函數（以 C 語言撰寫）來實現的。當 N<50 時，效率還好，當 N 過大時，可利用 FFT 來加速計算，作法如下：

$$x1(n)*x2(n)=IFFT[FFT(x1(n))* FFT(x2(n))]$$

＊ 高速區段旋積

先前 ovrlpsav 函數是以 DFT 發展得出，如今可利用基-2 FFT 取代 DFT 而得高速重疊保留演算法

```
function [y] = hsolpsav(x,h,N)
% High-speed Overlap-Save method of block convolutions using FFT
N = 2^(ceil(log10(N)/loh(2));
Lenx = length(x); M = length(h);
M1 = M-1; L = N-M1;
h = fft(h,N);
%
x = [zeros(1,M1), x, zeros(1,N-1)];
K = floor((Lexn+M1-1)/(L)); % # of blocks
Y = zeros(K+1,N);
```

```
for k=0:K;
    xk = fft(x(k*L+1:k*L+N));
    Y(k+1,:) = real(ifft(xk.*h));
end
Y = Y(:,M:N)'; y = (Y(:))';
```

註：以上程式執行有問題