신호와 시스템 프로젝트

2016170994 김다민

**Code 및 과정**

n = -3000:3000;

N = 400;

l=fix((length(n)+1)/2);

rectlength = 19; %rectlength를 19, 29 ,49로 바꾸어 가며 실행

% (1) 과정 노이즈를 생성하고 cos신호에 더해줌

noise = randn(1, size(n,2));

x = cos(2\*pi\*n/N)+ noise;

% (2) 과정 basic lowpass filter 만듬

h = zeros(1,length(n));

for i=l:l+ rectlength

h(1,i)=1;

end

% (3) 과정 교과서 과정대로 convolution 진행 후 y1에 저장

a = length(x);

b = length(h);

m = a + b -1;

y1 = zeros(1,m);

for i = 1:a

for j = 1:b

y1(i+j-1) = y1(i+j-1) + h(j)\*x(i);

end

end

y1\_s=y1./rectlength;

% y1\_s는 rect함수와 컨벌루션 후 스케일이 증가하기 때문에 원래 신호와 스케일이 같도록 rect함수의 길이로 나누어줌.

% (4) 과정 내장된 conv 함수를 이용하여 convolution실행후 y2에저장

y2 = conv(x, h);

y2\_s=y2./rectlength;

% y2\_s는 rect함수와 컨벌루션 후 스케일이 증가하기 때문에 원래 신호와 스케일이 같도록 rect함수의 길이로 나누어줌.

% (5) 과정 각 그래프를 범위를 조정해 비교하기 쉽게 출력

% 컨벌루션 진행 시 결과의 크기는 컨벌루션 두함수의 크기합-1로 늘어나고, 결과는 증가한 크기의 중앙에서 시작하므로 조정이 필요함.

tiledlayout(3,2)

nexttile

plot (x(1,l:l+2000))

nexttile

plot (h(1,l:l+100))

nexttile

plot (y1(1,2\*l+(rectlength+1)/2:2\*l+2000+(rectlength+1)/2))

nexttile

plot (y2(1,2\*l+(rectlength+1)/2:2\*l+2000+(rectlength+1)/2))

nexttile

plot (y1\_s(1,2\*l+(rectlength+1)/2:2\*l+2000+(rectlength+1)/2))

nexttile

plot (y2\_s(1,2\*l+(rectlength+1)/2:2\*l+2000+(rectlength+1)/2))

(5) 과정 결과

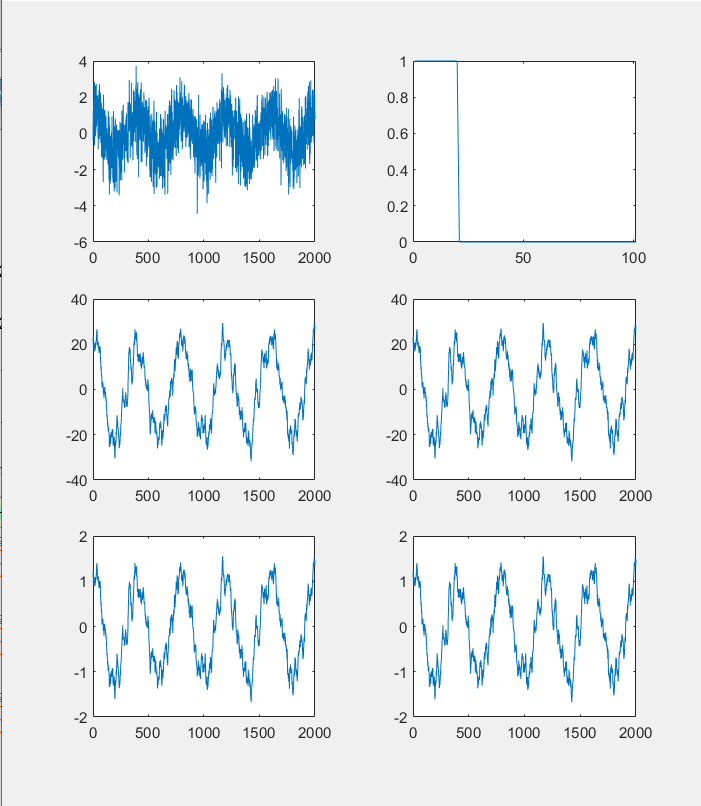
1.noise가 추가된 cos 신호, 2.rect function

3. 직접만든 conv결과 4. 내장함수 conv결과

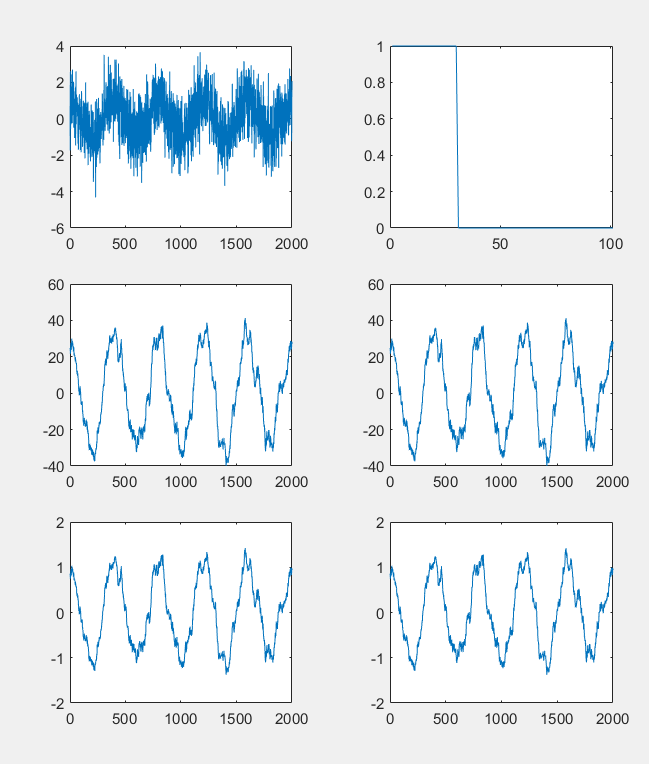
5. 직접만든 conv 스케일조정 6. 내장함수 conv 스케일조정

(rect함수의 길이가 길수록 원래함수의 최고 스케일이 rect함수의 길이배로 증가하므로 rect함수 길이로 나누어 스케일 조정함 rect(t/a) ↔ |a|\*sinc(af)이므로 |a|로나누어 스케일을 동등하게 맞춰봄)

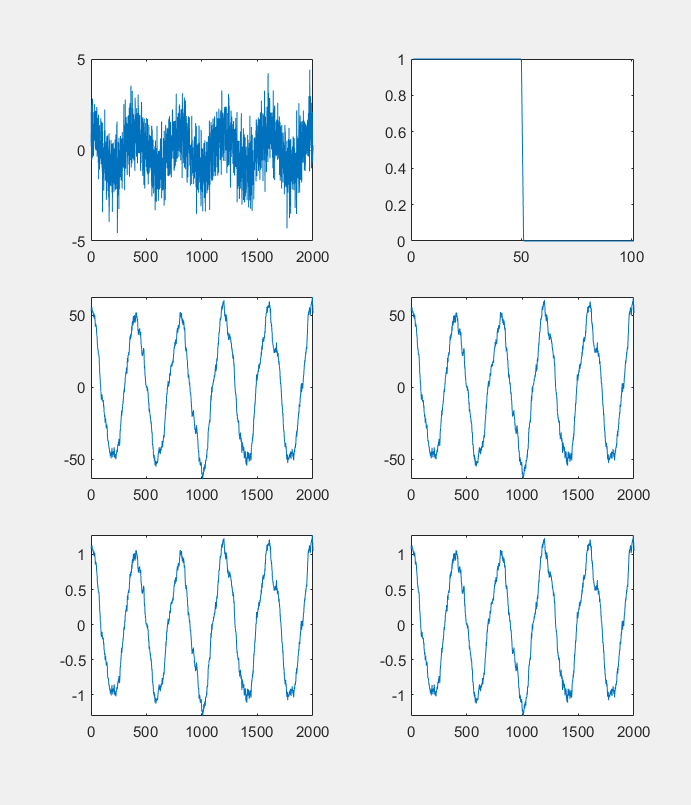
N=400 , rect함수 길이=19 일때



N=400 , rect함수 길이=29 일 때



N=400 , rect함수 길이=49 일때



(6) 메트랩의 교과서 과정으로 만든 convolution과 내장함수 conv의 차이점

Matlab 홈페이지에 있는 conv함수 예제

두 개의 벡터를 만들고 이 벡터를 컨벌루션합니다.

u = [1 1 1];

v = [1 1 0 0 0 1 1];

w = conv(u,v)

w = *1×9*

1 2 2 1 0 1 2 2 1

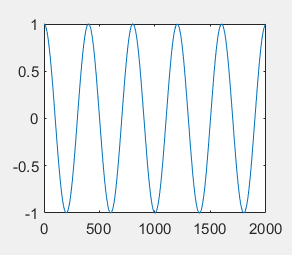
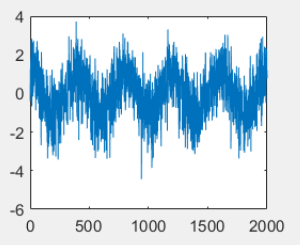
교과서 DT시스템 conv와 방식이 같았다.

메트랩 시스템 자체 행렬을 기반 으로한 DT system으로 구성 되어있다. 또한 메트랩 홈페이지에 나와있는 conv함수의 정의를 보았을 때 교과서에서 실행한 conv와 과정이 같았기 때문에 교과서의 DT conv와 메트랩의 conv와 차이가 나타나지 않았다.

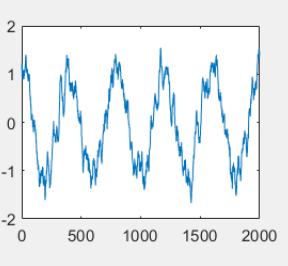
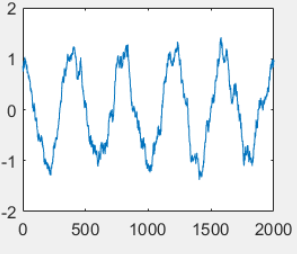
(7) Discuss/analyze the results

ℎ[𝑛] = 1, n = 0 … 19 ℎ[𝑛] = 1, n = 0 … 29 ℎ[𝑛] = 1, n = 0 … 49

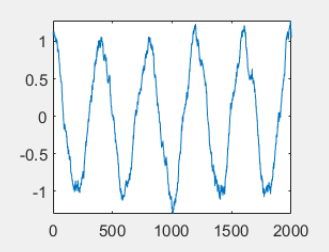
원래 cos 신호 노이즈가 낀 신호

rect함수 길이=19(스케일 조정됨) rect함수 길이=29 (스케일 조정됨)

rect함수 길이=49(스케일 조정됨)



위의 그림을 보면 알 수 있듯이 rect함수의 길이가 커질수록 원래 신호와 가까운 신호를 얻을 수 있었다.

원리를 파악해 보자면 rect함수를 푸리에 변환할 시 sinc함수 로 나타나는데

(프로젝트에서는 rect가 t가 전부 양수인곳에서 나타나기 때문에 t영역 이동을 해주어야 하지만, 이동을 하지않아도 원리가 같기 때문에 이동하지 않고 원리를 설명하겠습니다. )

rect함수의 길이가 길수록 푸리에 변환 다음 성질에 따라

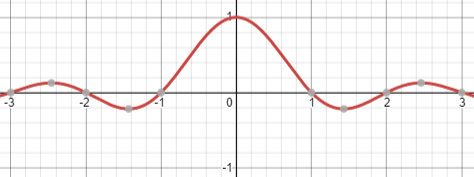
(a는 rect함수의 길이)

x(t/a) ↔ |a|\*X(af)

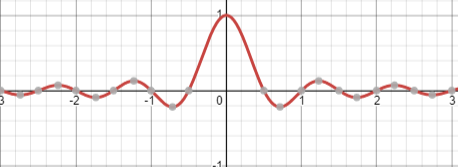
rect(t/a) ↔ |a|\*sinc(af)가 되므로 (sinc의 폭이 더좁아지는 형태로 나타난다.)

그래프로 나타내자면 푸리에 변환 시 |a|\*sinc(af)는 |a|배 만큼 싱크의 스케일이 증가하므로 스케일이 갖도록 1/|a|를 곱해주고 생각해보자. 즉, sinc(af)를 고려해본다.

rect길이가 1일 때 -> sinc(f)



rect길이가 2일 때 -> sinc(2f)



그래프에서 보면 알수 있듯이 rect의 길이가 길수록 sinc 의 폭이 좁아지는 걸 알 수 있는데 이는 rect의 길이가 길수록 저역대를 통과하는 비율을 높이고 고역대를 통과시키는 비율이 낮아짐을 의미한다. 따라서 rect가 증가 할수록 통과시키는 주파수대가 낮아진다고 할 수 있다.

이번 프로젝트의 경우 noise의 경우 랜덤 가우시안 변수로 범위내에서 빠르게 변화하는 고주파수 성분을 대부분 갖고 있기 때문에 LPF를 통과시 상대적으로 주파수가 낮은 신호인 cos신호가 두드러지게 나타날 수 있다. 따라서 코사인의 주파수(1/N)보다 LPF의 차단 대역대가 큰 영역에서는

rect의 길이가 길어질수록 cos신호보다 주파수가 높은 잡음 신호들이 제거되는 범위가 더 넓어지기 때문에 상대적으로 rect길이가 길수록 원래 신호와 비슷한 결과를 얻을 수 있다.

다만 rect의 경우가 너무 길어져 sinc함수의 폭이 너무 작아져 cos의 주파수 (1/N) 마저 통과시키지 못하는 경우 원래신호를 얻지 못하고 주파수성분이 매우 작은 잡음만 얻게 될 것이다.