

디지털통신시스템설계

Digital Communication System Capstone Design

ICE4009-001



6 주차 실습과제

정보통신공학과

12191765

박승재

Introduction

실습을 위한 Original Signal 을 생성한다.

```
% Original Signal
f_s = 1000; % sampling frequency
t_start = 0; % sampling start time
t_end = 10; % sampling end time

t_s = 1 / f_s; % sampling interval
t = t_start:t_s:t_end;

x = 2 * sin(4 * pi * t) + cos(6 * pi * t); % original signal
```

Sampling

Sampling 이란 continuous time 의 신호 $x(t)$ 를 discrete time 의 신호로 변환하는 것이다. Amplitude 측면에서는 여전히 continuous 성질을 가진다.

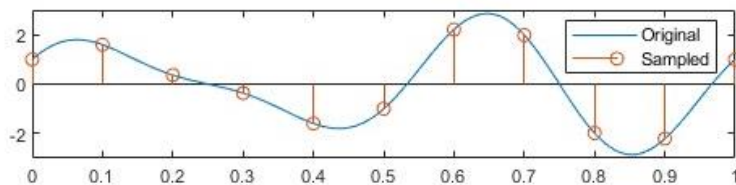
```
% Sampled Signal
f2_s = 10; % sampling frequency

t2_s = 1 / f2_s; % sampling interval
t2 = t_start:t2_s:t_end;

x2 = x(1:floor(t2_s / t_s):length(t)); % Sampled signal

subplot(3, 1, 1)
plot(t, x)
hold on
stem(t2, x2)
axis([0 1 -3 3])
legend('Original', 'Sampled')
```

샘플링 주파수 f_{2_s} 는 10 으로 설정했다.



파란색 Original Signal 를 샘플링하여 주황색 stem 그래프로 나타내었다.

Quantization

Quantization 은 continuous 한 amplitude 를 갖는 샘플링 신호를 discrete 한 amplitude 를 갖도록 amplitude 를 재구성하는 과정이다.

```
% Quantization
A_max = max(x2);
A_min = min(x2);
Q_level = 64;
Q_step = (A_max - A_min) / Q_level;

xq_level = min(floor((x2 - A_min) / Q_step), Q_level - 1);
xq = (xq_level * Q_step) + A_min;
```


xq =

1 ~ 10번 열

0.9674 1.5892 0.3455 -0.4146 -1.6583 -1.0365 2.1420 1.9347 -2.0038 -2.2111

11 ~ 20번 열

0.9674 1.5892 0.3455 -0.4146 -1.6583 -1.0365 2.1420 1.9347 -2.0038 -2.2111

 Q_step 0.0691

xq 가 Q_step 간격으로 양자화된 것을 확인할 수 있다.

Encoding

Encoding 은 양자화 신호를 2 진수의 신호로 변환한다. 양자화 레벨에 따라 하나의 샘플을 n-bit 의 신호로 변환한다.

```
tmp = dec2bin(xq_level);
xe = reshape(tmp, 1, numel(tmp));
```

tmp =

101×6 [char](#) 배열

```
'101110'
'110111'
'100101'
'011010'
'001000'
'010001'
'111111'
```

tmp 는 양자화된 값을 2 진수로 변환한 값이다.

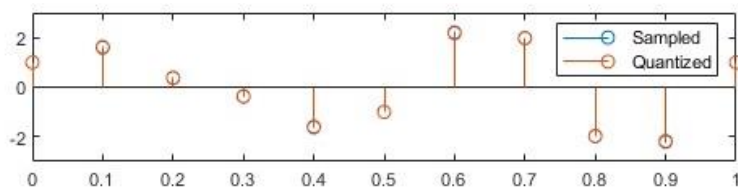
Decoding

Decoding 은 부호화된 신호를 양자화 신호로 변환하는 과정이다.

```
tmp2 = reshape(xe, length(t2), log2(Q_level));
xd = Q_step * (bin2dec(tmp2) - (Q_level / 2) + 0.5);

subplot(3, 1, 2)
stem(t2, x2)
hold on
stem(t2, xd)
axis([0 1 -3 3])
legend('Sampled', 'Quantized')
```

xd 에 0.5 를 더한 이유는 양자화 레벨의 가운데 값으로 맞춰주기 위함이다.



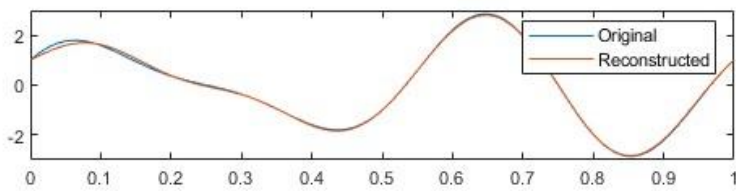
Original Signal 와 Quantization → Encoding → Decoding 을 거친 Signal 을 비교해보면 거의 오차가 나지 않는 것을 확인할 수 있다. 여기서 발생한 오차를 양자화 잡음이라 한다. `Q_level` 이 커진다면 오차는 더 줄어들 것이다.

Reconstruction

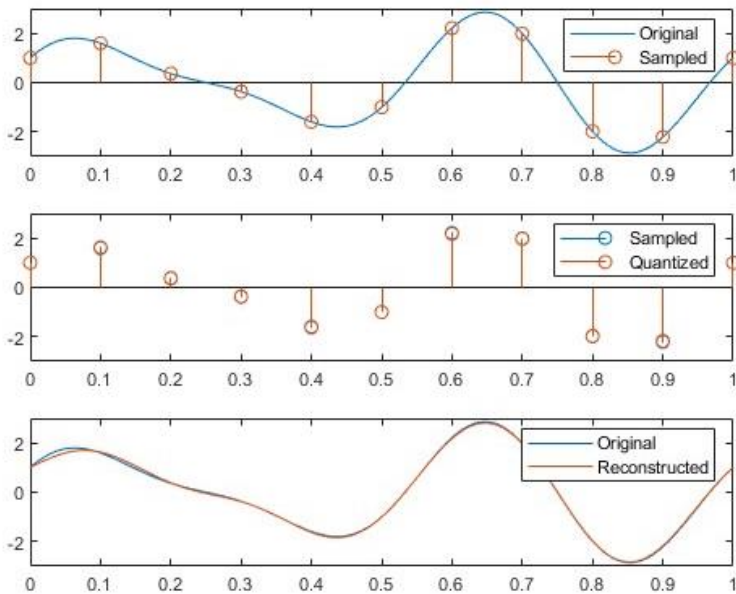
복호화된 신호에 sinc 함수를 곱해 원래의 신호를 복원한다.

```
% Reconstructed Signal
y = zeros(length(t2), length(t));
for idx = 1:length(t2)
    y(idx, :) = xd(idx) * sinc((t - (t(1) + (idx - 1) * t2_s)) / t2_s);
end
y = sum(y);

subplot(3, 1, 3)
plot(t, x, t, y)
axis([0 1 -3 3])
legend('Original', 'Reconstructed')
```



Original Signal 과 비교해보면 약간의 차이가 있지만 거의 오차가 발생하지 않는 것을 확인할 수 있다. 샘플링 주파수와 양자화 레벨을 올릴수록 오차는 줄어들 것이다.



위는 최종 그래프 출력 결과이다.

Problem

제공된 부호화 신호를 복원하고 wav 파일로 재생하시오

- 부호화 신호 읽어오기 : `load('encode_data.mat')` → `x_en` 생성
- 신호 정보 :
 - 신호의 길이 : 5.5 초 ($t = 0 : 1/f_0 : 5.5$)
 - 원본 wav 파일의 샘플링 주파수 : $f_0 = 22.05$ kHz
 - 샘플링 신호의 샘플링 주파수 : $f_s = 11.025$ kHz
 - 양자화 레벨 : 64

- 신호 최대값: 1, 신호 최소값 : -1
- WAV 파일 생성 관련 → 매트랩 함수 audiowrite 참고
- 파일을 복원하고, 재생해서 내용에 대해 작성하시오.

Result

```
% Original Signal
f_s = 22050; % sampling frequency
t_start = 0; % sampling start time
t_end = 5.5; % sampling end time

t_s = 1 / f_s; % sampling interval
t = t_start:t_s:t_end;

% Sampled Signal
f2_s = 11025; % sampling frequency

t2_s = 1 / f2_s; % sampling interval
t2 = t_start:t2_s:(t_end - t2_s);

% Load x_en
load('encode_data.mat')

% Quantization
A_max = 1;
A_min = -1;
Q_level = 64;
Q_step = (A_max - A_min) / Q_level;

% Decode
tmp2 = reshape(x_en, log2(Q_level), length(t2))';
x_de = zeros(1, length(t2));
for idx = 1:length(t2)
    x_de(idx) = Q_step * (bin2dec(tmp2(idx, :)) - (Q_level / 2) + 0.5);
end

% Reconstructed Signal
y = zeros(1, length(t));
for idx = 1:length(t2)
    y = y + (x_de(idx) * sinc((t - (t(1) + (idx - 1) * t2_s)) / t2_s));
    progress = idx / length(t2) * 100
end
```

```
audiowrite('data.wav', y, f_s);
```

data.wav: 다가오는 주말에는 전국에 비가 오면서 공기가 더 깨끗해질 전망입니다.

`load('encode_data.mat')`을 통해 `x_en`을 가져오기 때문에 기존의 Sampling과 Encoding 코드를 삭제한다.

```
x_en
1x363822 char

val =

'100000100000100000100000100000100000100000111110111110111111'
```

`x_en`을 받아서 `reshape`해 `tmp2`를 만든다.

```
tmp2
60637x6 char

val =

'100000'
'100000'
'100000'
'100000'
'100000'
'100000'
'100000'
'100000'
'011111'
'011111'
```

이진수로 표현된 양자화 레벨을 원래 값으로 변환한다.

```
x_de
1x60637 double



|   | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8       | 9       | 10      | 11     |
|---|--------|--------|--------|--------|--------|--------|--------|---------|---------|---------|--------|
| 1 | 0.0156 | 0.0156 | 0.0156 | 0.0156 | 0.0156 | 0.0156 | 0.0156 | -0.0156 | -0.0156 | -0.0156 | 0.0156 |


```

샘플링한 값인 `x_de`가 추출되었으니 `sinc` 함수를 곱해 원래의 신호를 복원하면 된다.

```
y
1x121276 double



|   | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     |
|---|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 1 | 0.0156 | 0.0159 | 0.0156 | 0.0165 | 0.0156 | 0.0140 | 0.0156 | 0.0180 | 0.0156 | 0.0122 | 0.0156 |

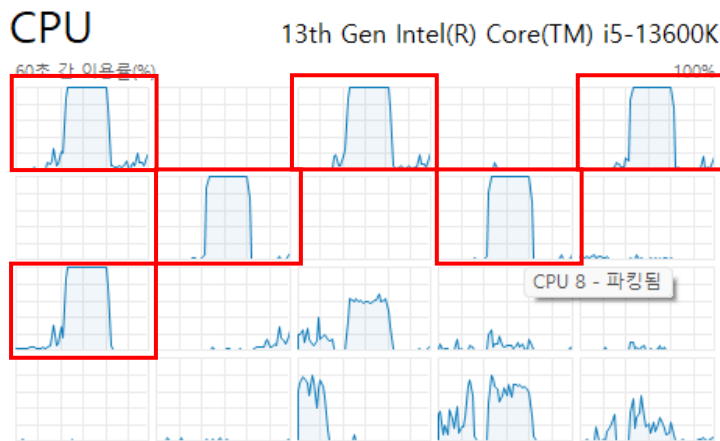

```

기존의 Reconstruction 코드를 이용해 복원하려 시도하면 아래와 같은 오류가 발생한다.

```
다음 사용 중 오류가 발생함: zeros
요청된 60637x121276 (54.8GB) 배열이 최대 배열 크기 기본 설정 (31.8GB)을 초과합니다. 이로 인해 MATLAB이 응답하지 않을 수 있습니다.

오류 발생: decode_data (27번 라인)
y = zeros(length(t2), length(t));
```

메모리 부족 문제를 해결하기 위해 한 번에 `sum`을 하는 대신, `y`를 `zeros(1, length(t))`로 잡고 각 반복마다 `y`에 값을 더하는 식으로 구현했다. 계산이 오래 걸리는 작업이기 때문에 `progress = idx / length(t2) * 100`를 넣어 계산이 100% 완료되기까지 얼마나 남았는지 출력하도록 했다.



신호를 복원하는 중에는 CPU 점유율이 올라간다. 계산에는 6 개의 CPU 코어가 사용되는 것 같다.

```

명령 창
MATLAB을 처음 사용한다면 시작하기를 참조하십시오.
99.9967

progress =
    99.9984

progress =
    100

fx >> |
  
```

progress 가 100 이 되면 계산이 끝나면서 `audiowrite` 가 호출되며 `data.wav` 파일이 만들어진다. wav 파일은 "다가오는 주말에는 전국에 비가 오면서 공기가 더 깨끗해질 전망입니다."라고 말하는 여자 아나운서 목소리를 출력한다.

Conclusion

Sampling → Quantization → Encoding → Decoding → Reconstruction 실습을 수행했다. `dec2bin` 와 `bin2dec` 를 이용해 데이터를 이진수로 Encoding/Decoding 하는 방법을 배웠고, 과제를 하면서 대용량의 데이터를 복원하는데 메모리를 적게 사용할 방법을 생각하게 되었다.

지난 실습에서 추가된 내용은 Quantization, Encoding, Decoding 밖에 없지만 이 과정이 생각보다 복잡하고, 직접 실습을 하면서 데이터가 어떻게 변환되는지 확인할 수 있었다. 실습에서 Original 과 Reconstructed 신호의 오차는 샘플링과 양자화 잡음에서만 발생된다. (Channel 을 거치며 신호에 잡음이 추가되지 않기 때문에) 따라서 샘플링 레이트를 올리고 양자화 레벨을 높인다면 오차는 줄어들 것이다. 하지만 샘플링 레이트와 양자화 레벨을 무작정 높인다면 그만큼 보내야할 데이터가 늘어나기 때문에 현실에서는 오차와 전송량의 타협점을 찾아야할 필요가 있다.

과제에서는 샘플링 주파수로 11.025kHz 를 사용했는데 사람의 가청주파수인 20kHz 보다는 낮지만, 목소리 주파수(3.4kHz)의 2 배인 6.8kHz 이상으로 샘플링하여 충분히 소리를 구분할 수 있었다.

목소리 주파수(3.4kHz)의 2 배 이상으로 샘플링해야 하는 이유는 Nyquist Theorem 에 의해 신호의 최대 주파수의 2 배 이상으로 샘플링해야 원래의 신호를 복원가능하기 때문이다. Nyquist Theorem 과 관련해서는 지난 과제에서 이미 실습한 바 있다.