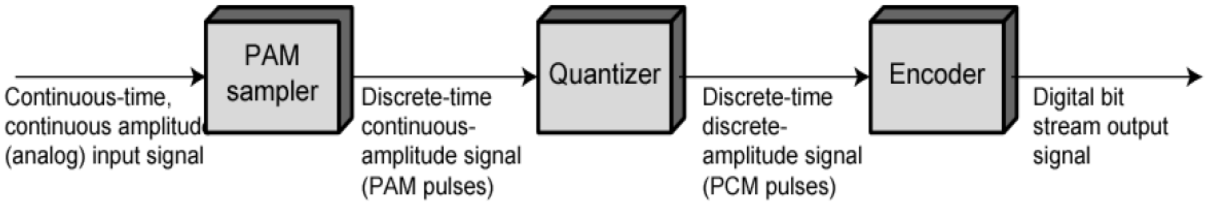
CS2033 Data Communication and Networks – In20

Department of Computer Science and Engineering
University of Moratuwa

Lab Assignment 2 – Pulse Code Modulation

We learned about Pulse Code Modulation (PCM), a technique that is used to convert an analog signal to a digital signal. In this lab you will implement a Pulse Code Modulator as a C program.

The following figure shows the main components involved in PCM.


Continuous-time, continuous amplitude (analog) input signal
PAM sampler
Discrete-time continuous-amplitude signal (PAM pulses)
Quantizer
Discrete-time discrete-amplitude signal (PCM pulses)
Encoder
Digital bit stream output signal

In this lab, you will implement the Pulse Code Modulator in a modular way where each module will be responsible for a specific PCM function. Moreover, to test and showcase the operation of your PCM program, you will need to implement a simple analog signal generator. Thus, the following modules need to be implemented.

1. Analog Signal Generator
2. Sampler
3. Quantizer
4. Encoder

Rest of this document provides instructions regarding implementing these four components.

Lab 2.1 Analog Signal Generator

```
float analog_signal_generator(asignal signal, int t)
```

where asignal is defined (in "mypcm.h") as follows;

```
typedef struct {
  float A;
  float omega;
  float sigma;
  int duration;
} asignal;
```

For simplicity, we assume only a sinusoidal analog will be used for this lab. As you should already know, any sinusoidal signal can be represented with

$$x(t) = A\sin(\omega t + \sigma)$$

Where t is time, A is the peak amplitude, ω is the angular frequency in radians, and σ is the phase in radians. duration is the number of time units that the signal exists.

The analog signal generator function will take as input `A, omega, sigma,` and `t`, and output the signal value `x(t)` at time `t` given the input values.

Lab 2.2 Sampler

void sampler(float *samples, int interval, asignal signal)

The sampler function takes as input a pointer to an array where samples will be stored, the sampling interval and the analog signal and updates the samples array with the relevant sample values.

Lab 2.3 Quantizer

void quantizer(float *samples, int *pcmpulses, int levels)

The quantizer function takes as input a pointer to the samples array, a pointer to the pcmpulses array where the output of quantizer will be stores, and the quantization levels, and produces PCM pulses accordingly.

For simplicity, in this lab assignment, please take the "floor" value as the quantized value. Quantization logic can be expressed with the following formula:

$$Q_{level}(t) = \left\lfloor \left(\left(\frac{x(t) - A_{min}}{A_{max} - A_{min}} \right) \times Number\ of\ Quantization\ Levels \right) \right\rfloor$$

When $x(t) = A_{max}$ the quantization level should be set to maximum quantization level.

Lab 2.4 Encoder

void encoder(int *pcmpulses, int *dsignal, int encoderbits)

The encoder takes as input a pointer to pcmpulses array, a pointer to dsignal array where output will be stored, and the number of bits used for a single PCM code. It produces the PCM codes for each pulse recorded and stores in dsignal as a **binary number stream**.

The void run() function

In addition to the main components, you need to have a run() function that does the following:

1. Reads the following input parameters from the **stdin**. These will be given as **space-separated** values.

   ```
   A omega sigma duration interval encoderbits
   ```

2. Prints the final output as a bit stream (consisting of 1s and 0s) to the **stdout.**
3. Perform any other auxiliary tasks, if necessary.

Example

Suppose the analog signal is x(t) = **`3 sin(2t + 0).`**

Also assume that the signal **`duration = 5`**

Given these facts a possible input string is:

```
3 2 0 5 1 2
```

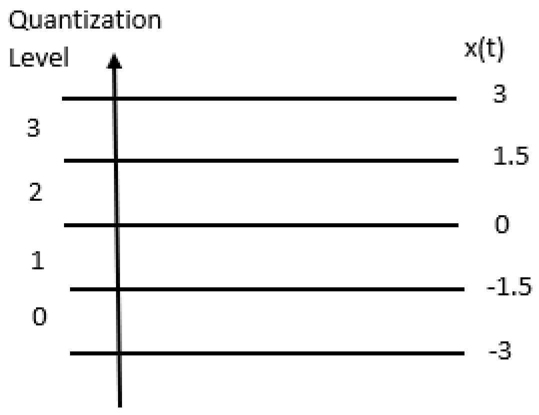
In this input string

A = 3
omega = 2
sigma = 0
duration = 5
interval = 1
encoderbits = 2

Thus, you should generate the sine wave using the give values, take samples every 1 time unit, assign samples into one of 4 quantization levels (since we are using 2 encoderbits as per the input string), and produce the encoder output bitstream. Please note that the quantization levels should span from the minimum peak to the maximum peak. In other words, quantization levels should cover both -3 as well as +3. In this example, quantization level 0 should start at -3 and, quantization level 3 should correspond to +3.

The samples from t=0 to t=5 will be as follows.

When t = 0; x(0) = 0
When t = 1; x(1) = 2.73
When t = 2; x(2) = -2.27
When t = 3; x(3) = -0.84
When t = 4; x(4) = 2.97
When t = 5; x(5) = -1.63



Calculating quantization levels:

$$Q_{level}(t) = \left\lfloor \left(\left(\frac{x(t) - A_{min}}{A_{max} - A_{min}} \right) \times Number\ of\ Quantization\ Levels \right) \right\rfloor$$

When t = 0; x(0) = 0

$$Q_{level}(0) = \left\lfloor \left(\frac{0-(-3)}{3-(-3)} \right) \times 4 \right\rfloor = 2$$

When t = 1; x(1) = 2.73

$$Q_{level}(1) = \left\lfloor \left(\frac{2.73-(-3)}{3-(-3)} \right) \times 4 \right\rfloor = 3$$

When t = 2; x(2) = -2.27

$$Q_{level}(2) = \left\lfloor \left(\frac{-2.27-(-3)}{3-(-3)}\right) \times 4 \right\rfloor = 0$$

When t = 3; x(3) = -0.84

$$Q_{level}(3) = \left\lfloor \left(\frac{-0.84-(-3)}{3-(-3)}\right) \times 4 \right\rfloor = 1$$

When t = 4; x(4) = 2.97

$$Q_{level}(4) = \left\lfloor \left(\frac{2.97-(-3)}{3-(-3)}\right) \times 4 \right\rfloor = 3$$

When t = 5; x(5) = -1.63

$$Q_{level}(5) = \left\lfloor \left(\frac{-1.63-(-3)}{3-(-3)}\right) \times 4 \right\rfloor = 0$$

Thus, the PCM encoded output binary stream for the given analog signal is:

```
101100011100
```

Please try out different analog signals by changing the input parameters test the correctness of your program before submitting.

Submission Instructions

You have been provided with a code template that consists of the following files:

lab2.c
mypcm.c
mypcm.h
Makefile

Please modify mypcm.c and lab2.c to complete the program.

You should submit the completed source code into the coderunner autograded quiz on moodle like you did for lab1.

Note: since this is a brand-new lab designed for your batch, there may be glitches. If you come across any such glitches, kindly inform me (sunimal@cse.mrt.ac.lk).

Please note that this is an individual assignment, and those who are caught plagiarizing will be penalized and will be reported for disciplinary action.