Quantization and analog-to-digital conversion

Due: Saturday, 29 February 2020, 11:55 PM (Valid only for Moodle submissions. Oral defense should be done before this date.)

Submission Format: Report (PDF, the report should contain your goal, steps you took to achieve it and outcomes, conclusion) and source file in Moodle.

Oral defense: Will take place primarily during class on Week 6, with extra slots during office hours up until the end of Week 6.

Introduction

More than one hundred years ago, the poet wrote a poem. We will try to "record" that poem.

Sound is a vibration that propagates as an acoustic wave. When we hear the speech, the acoustic wave propagates to our ear, and we notice the change in our eardrum's position. A different way to register that change in the air pressure, is to use some membrane with a copper coil or another sound-sensitive device, which can convert sound into an electrical signal (a microphone).

However, even we have the electrical signal that corresponds to our speech, yet our computer can not manipulate this information. The sound changes continuously. Continuous signals are often called analog signals. To convert the analog signal into digital, we must measure when and how the voltage was changed.

We cannot detect the change of voltage with 100% accuracy, but we can catch the discrete-time when the voltage reached some preset level. This process is known as quantization: mapping input values from a large set (often a continuous set) to output values in a (countable) smaller set [wikipedia].

In the electronics, for such quantization, analog-to-digital converters (ADCs) might be made of voltage comparators with the Priority encoder (fig. 1). When the input voltage from the microphone is equal to the voltage level on the comparator, the `True` value is on the comparator's output. The Priority encoder encodes all the data from other voltage level comparators into digital number form. The Priority encoder on this scheme works only at discrete-time when CLOCK input is `True`.

We will use the simulated model of such ADC. But with one assumption, as if we can change the number of comparators and threshold voltage levels in the circuit.

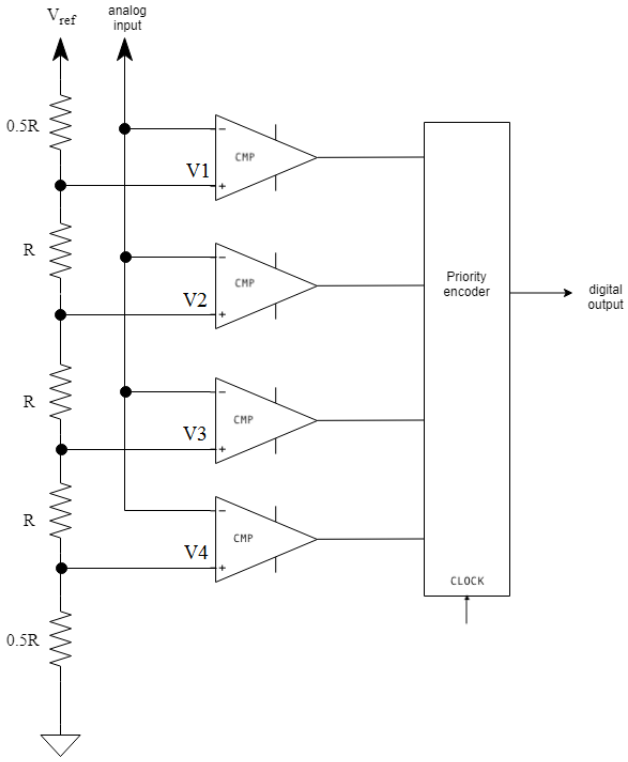

Figure 1. Example of ADC. Analog signal from the microphone compares with V1 ... V4 voltage levels of quantization on the comparators (CMP) input. Four levels of quantization might be decoded with 2-bit resolution.

Lab Assignment

Preparations

1. Download and unzip the archive

2. Launch Scilab, create a new file and set the unzipped folder as the working directory

```
b = chdir(path)
```

where 'path' is for example, `C:\Users\username\Downloads\lab_2` Scilab's chdir help

3. Connect the ADC simulator to the Scilab

`ADC.sce` file from the archive is the ADC simulator. We will follow the task with it. It takes data from the `data` folder from the achive; thus, it should be in the same directory. Load ADC function:

```
exec('ADC.sce')
```

Scilab's exec help

Task

1. Pick correct quantization levels and the sampling rate

Try to analyze your ADC and its output data. Choose levels of quantization and sampling rate fs (samples per second), until you hear the poet's voice as clear as possible.

2. Eliminate amplitude shift

When you can hear the poet's voice clearly, look at the waveform, you must notice a positive or negative shift from zero. Compensate it by adding or subtracting the value of that shift.

3. Eliminate sinusoidal interference

Our microphone also recorded some sinusoidal noise. We know only that the frequency of noise is from 120 to 210 Hz. Noise amplitude is 0.1, and the phase shift is 0. To compensate for the noise, subtract the sine function from the recorded signal.

Additional Task

Guess the author and merge the whole poem.

Tutorial

Choosing the quantization levels and sampling rate

Quantization levels limit the resolution of the signal. Quantization is the process that transforms real-valued signal into the signal that takes values only from a predefined discrete set. Normally, quantization levels change with a fixed increment (but not always). The output value after quantization could never be between two adjacent quantization levels. Let our chosen quantization levels be -1, 0 and 1, and the sampling rate (the rate a which we measure the voltage at the microphone) is 100 times per second:

```
fs = 100
quant_levels = [-1 0 1]
recorded_data = ADC(n, quant_levels, fs)
```

`n` - student number from 1 to 14 (ask your TA). Next, look at the plot of the recorded data. 100 samples represent one second of the recording. We limit amplitude value between -2 and 2.

```
f = figure(1) // set figure's number
clf // clear figure
plot(recorded_data, '--o')// plot data with o-samples
gca.data_bounds = [0,-1;fs,1] // x_limit 0...100, y_limit -2...2
xlabel('Samples')
ylabel('Amplitude')
```

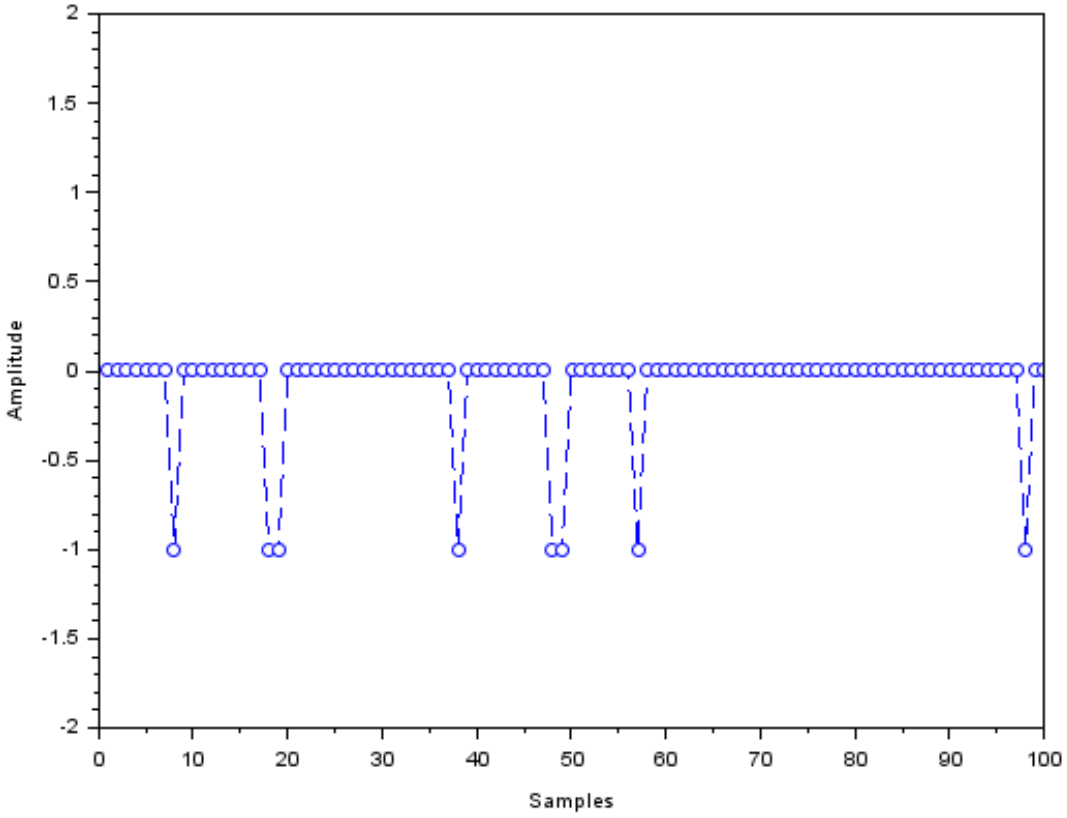

Figure 2. One second of the recorded data from the first iteration.

As we can see from the previous figure, some samples have value -1 and others - 0. Let us try other quantization levels and sampling rates.

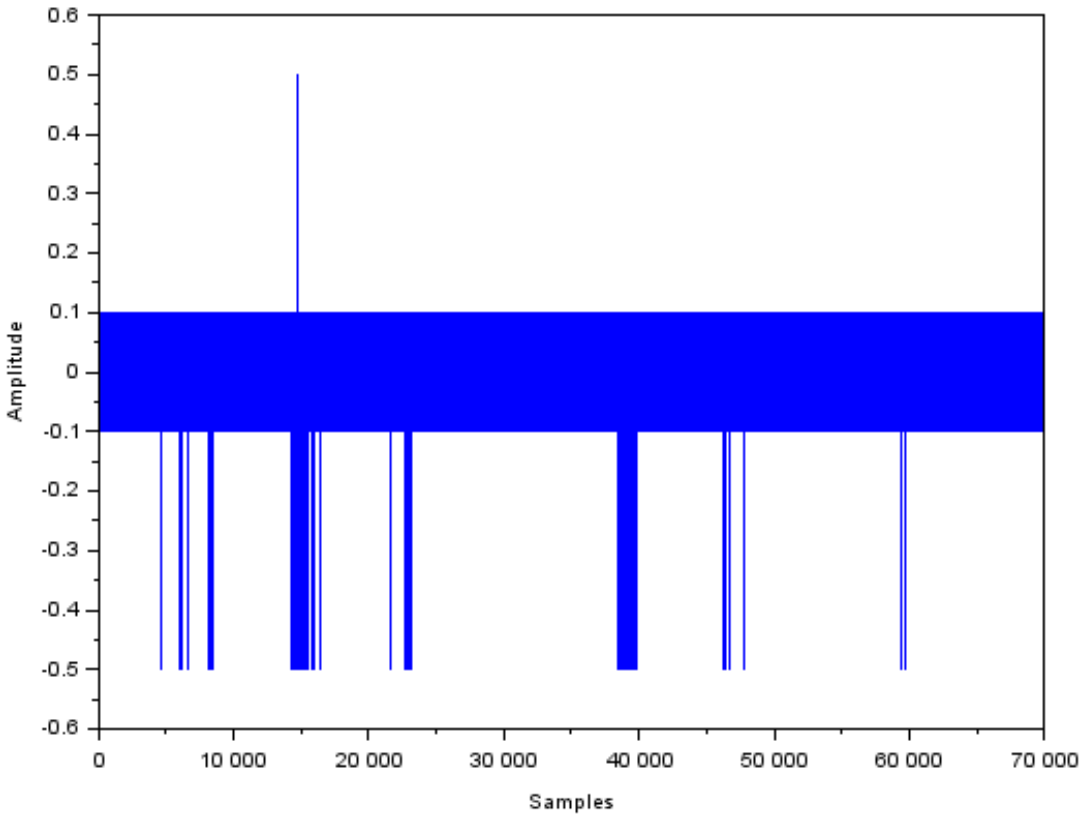

Figure 3. Output with quantization levels [-0.5, -0.1, 0.1, 0.5]; 10 000 samples per second

This time, we have more information about the recording. Samples primarily have values -0.5, -0.1, and 0.1. With this sampling rate, we can hear the voice in the recorded_data when played back with by Scilab's `playsnd`:

```
playsnd(recorded_data, fs)
```

To complete the task 1, change quantization levels and sampling rates, listen to the discretized sound until you hear the poet's voice as clear as possible.

Sinusoidal signal generation and signal cancellation

Suppose we have a signal with some added noise. In our case, the role of the noise is played by a sinusoidal signal. We can remove this noise by subtracting the original sinusoidal signal from the recorded signal. The sinusoidal signal has such parameters as amplitude A, frequency f, and phase offset ϕ.

The equation for sinusoidal signal is

$$y_{sin}(x) = A \cdot sin(2\pi f x + \phi)$$

For the sake of simplicity, we assume that $\phi = 0$. Since we are dealing with a discrete signal, we should produce values only at descrete momens of time.

$$y_s(n) = A \cdot sin(2 \cdot \pi \cdot f \cdot \Delta t \cdot n)$$
$$= A \cdot sin(2 \cdot \pi \cdot \frac{f}{f_s} \cdot n)$$

where f_s is the sampling rate (Hz or $Samples/Sec$). The argument of the sin function, i.e. $2 \cdot \pi \cdot \frac{f}{f_s} \cdot n$, is called *phase*. Let us first practice in plotting a sinusoidal function. Given the set of phases 0, $\pi/2$, π, $3\pi/2$, and 2π, we produce a figure.

```
samples = [0,%pi/2,%pi,3*%pi/2, 2*%pi]

clf
plot(sin(samples), '--o')
xlabel('Samples')
ylabel('Amplitude')
```

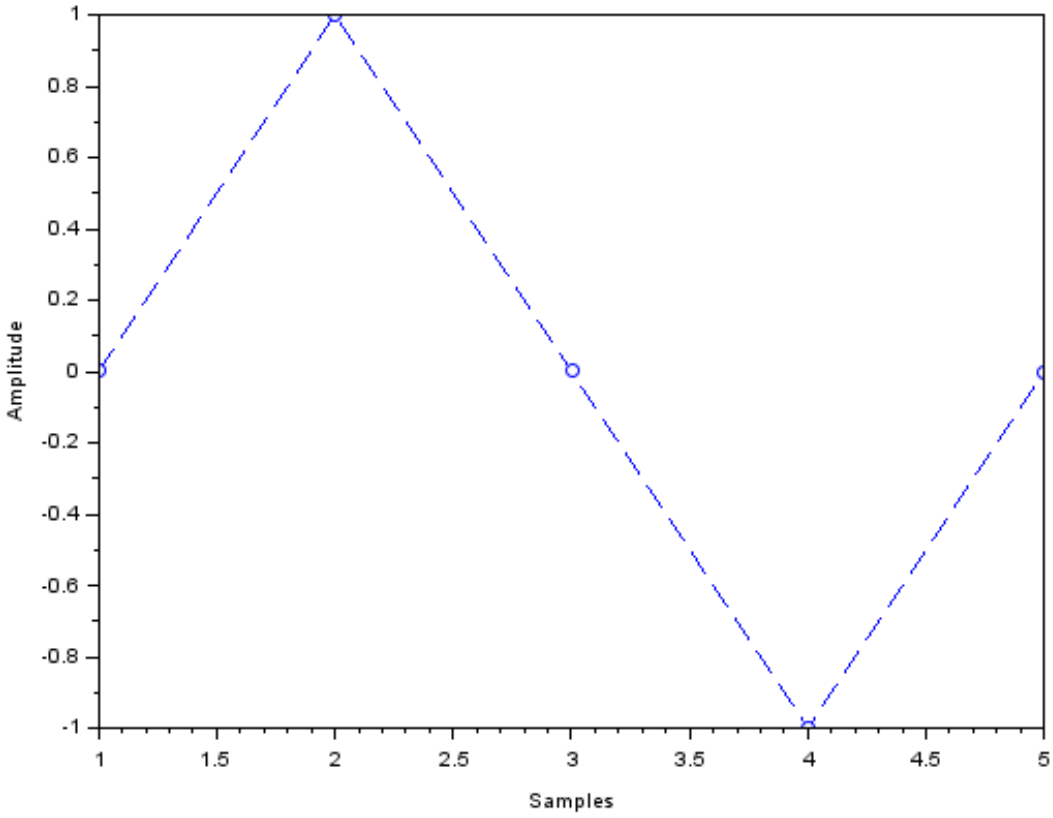

Figure 4. Result of a sine function

Now, assume that sampling rate of our recorded signal is, for instance, $100\ Samples/Sec = \frac{100 Samples}{1 Sec} = 100 Hz$. We want to build the sinusoidal wave with the frequency of $1 Hz$. In this case, one second of the sinusoidal signal must contain 100 samples. The function that generates such a signal is

$$y_s(n) = 1 \cdot sin(2 \cdot \pi \cdot \frac{1Hz}{100Hz} \cdot n)$$

The size of a step between phases should be equal to $\frac{2\pi \cdot 1}{fs}$.

```
fs = 100 //samples per second
step_size = (2*%pi)/fs;
samples = [1:fs]*step_size;
```

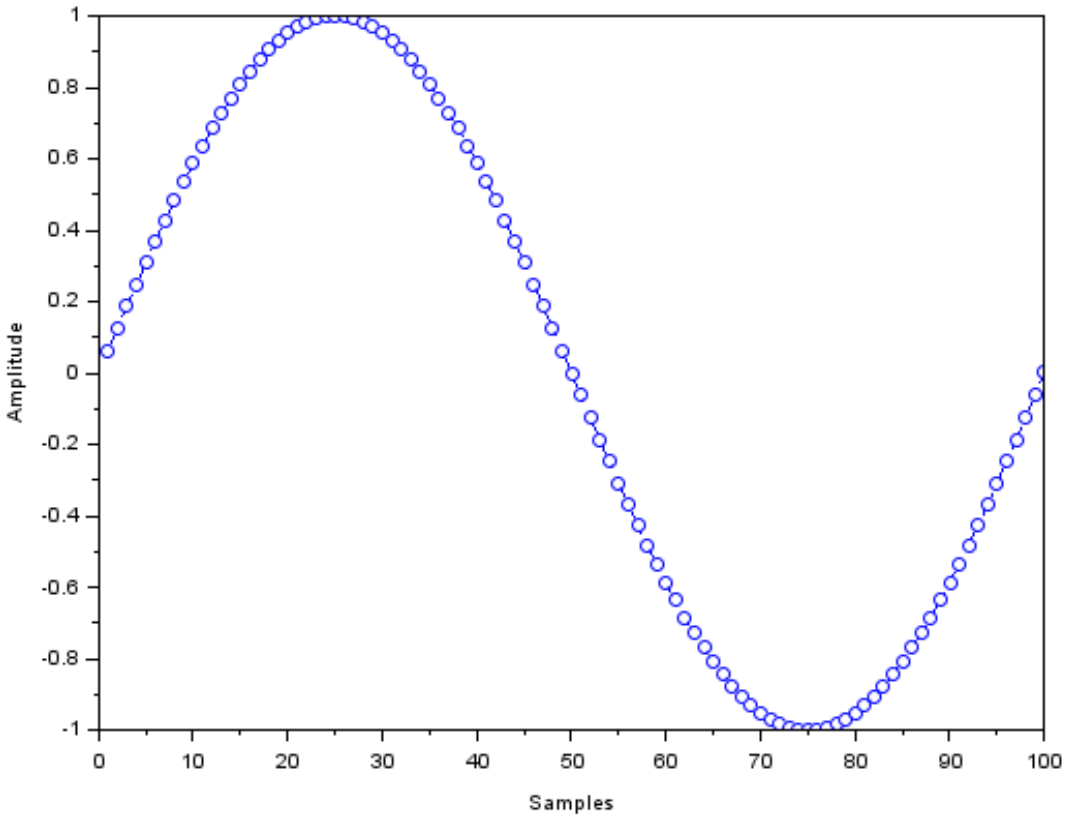

Figure 5. 100 samples, 1Hz sinusoidal signal. The sample rate is the same as our chosen.

Lastly, assume the frequency of the sinusoid, for instance, 2Hz, and the amplitude, is 0.5. The equations is

$$y_s(n) = 0.5 \cdot sin(2 \cdot \pi \cdot \frac{2Hz}{100Hz} \cdot n)$$

```
sin_freq = 2
sin_ampl = 0.5
fs = 100;
step_size = sin_freq*(2*%pi)/fs;
samples = [1:fs]*step_size;

clf
plot(sin_ampl*sin(samples), '--o')
xlabel('Samples')
ylabel('Amplitude')
```

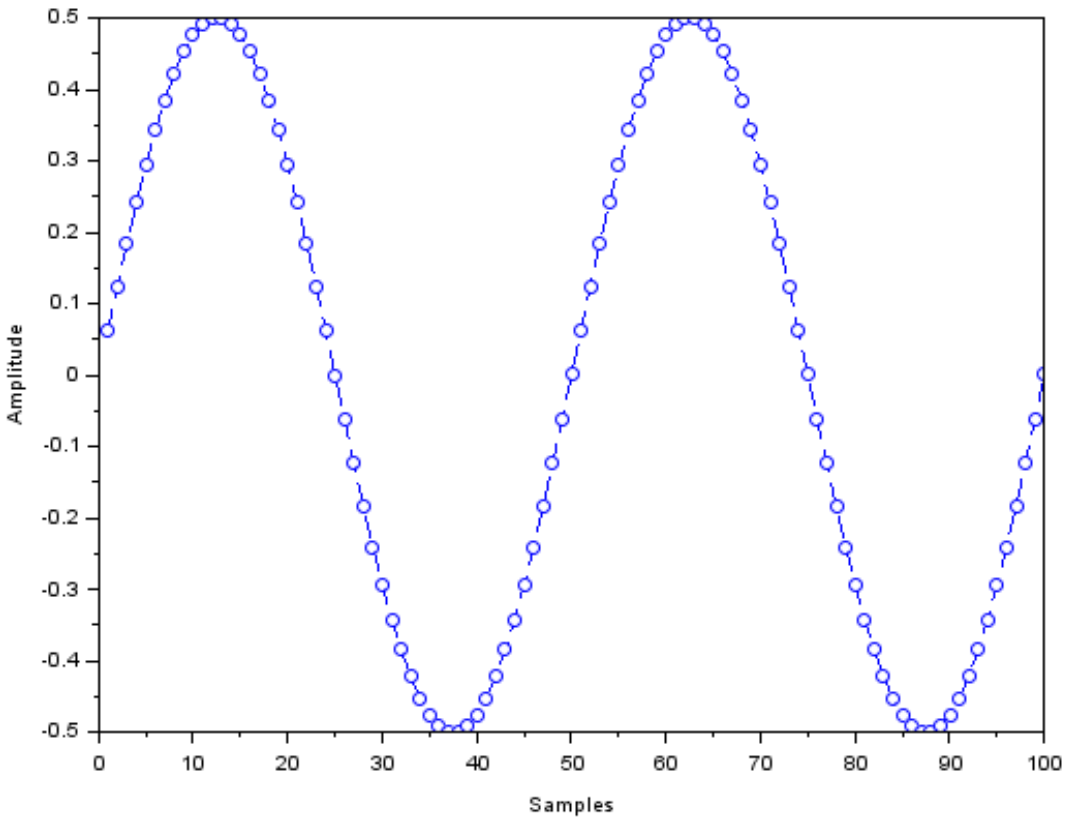

Figure 6. 100 samples, a 2Hz sinusoidal signal with an ampliude 0.5.

The example of a recorded signal with sinusoidal noise is shown in Figure 7. The recording length is one second. The sampling rate is $1000\ Samples/Sec$. In that case, if we count the number of main sine periods during one second, we can see that frequency is 5Hz, the amplitude is about 0.5.

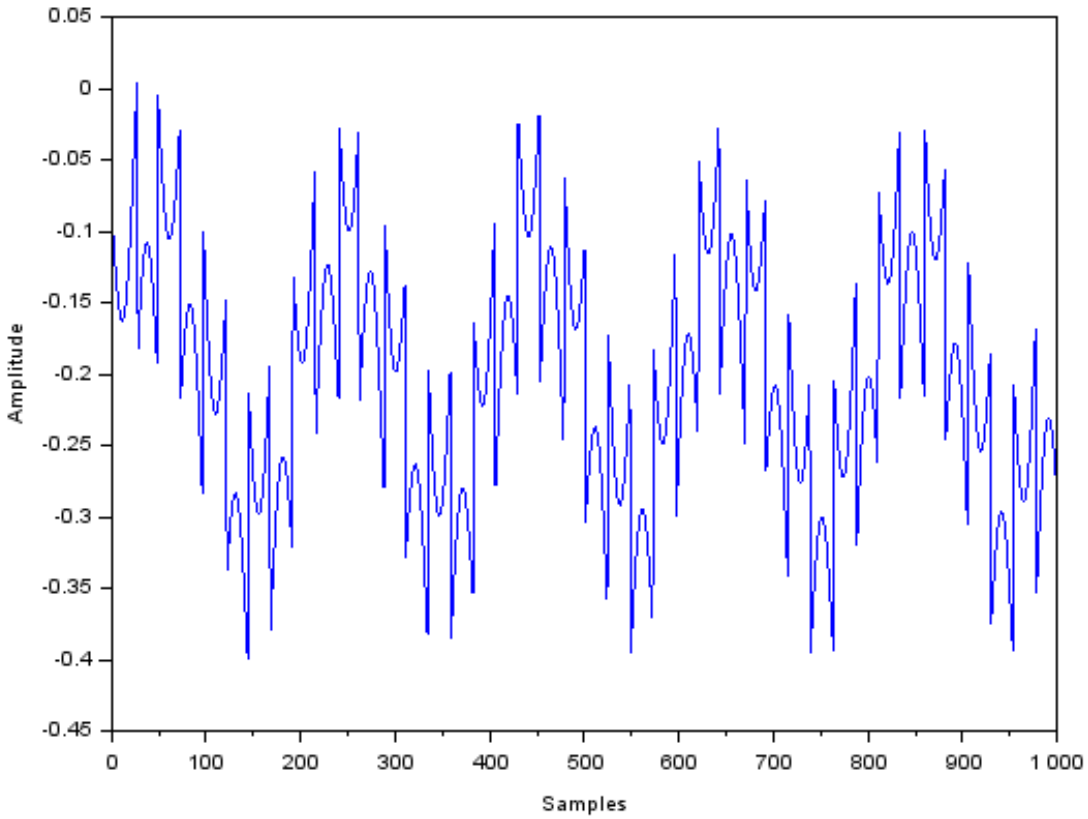

Figure 7. One second of recorded signal with a sine noise.

By subtracting such sinusoidal wave from the recording,

```
...
sin_sig = 0.5*sin(samples)

clf
plot(recorded_data - sin_sig)
```

we see the data with the noise canceled in Figure 8.

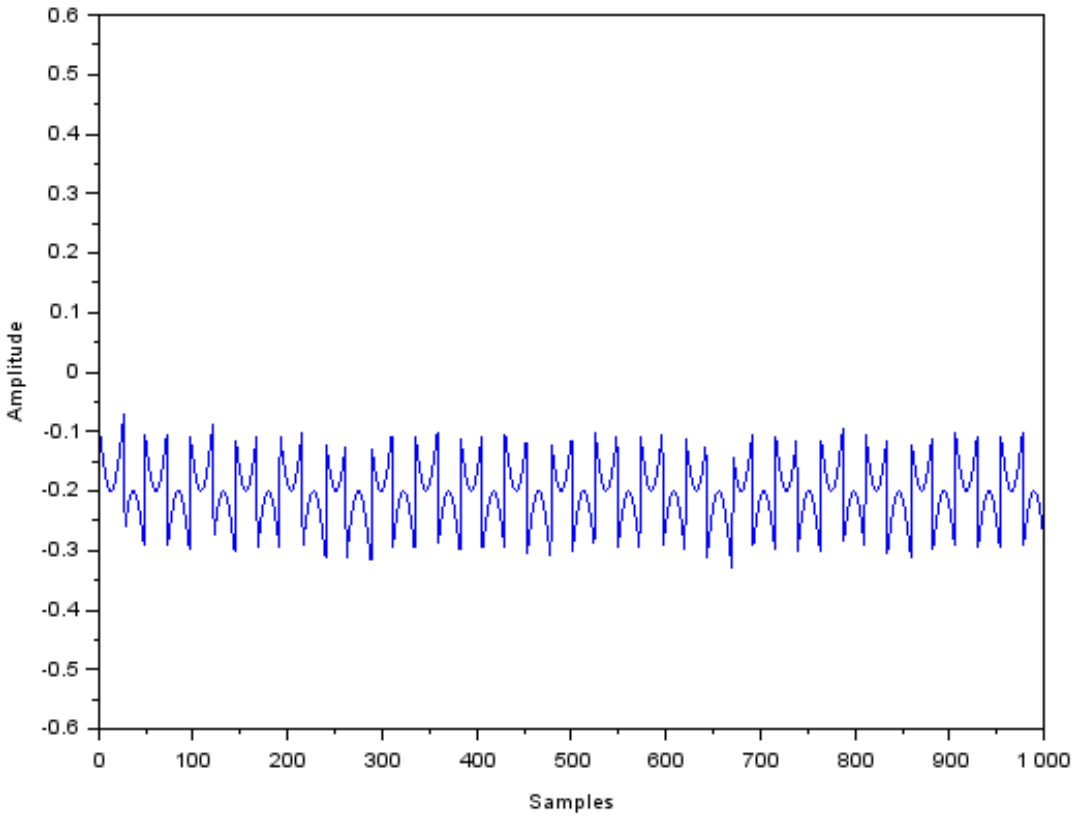

Figure 8. Cleared data.

To complete the task 3, subtract sinusoidal wave from the recording. The range of possible frequencies is from 120 to 210 Hz. The amplitude value is 0.1.

Self-Check Questions

- What is the sampling rate?
- How does one choose the sampling rate?
- What is quantization?
- How does one choose quantization levels?
- Define a linear system. Describe the superposition principle.