Name: David, Ruzzel S.
Section: CPE-401
Date Assigned:
Date Submitted:August 29,2023

EXPERIMENT 3
Sampling of Continuous Time Signal

OBJECTIVES

1. To simulate a continuous time (CT) signal.
2. To investigate the effect of sampling a CT signal.
3. To investigate some mathematical operations on DT signals.

MATERIALS AND EQUIPMENT

Computer installed with Matlab R2018 or Octave 5.2.0

AUDIO FILES

speech_dft.wav , flute.wav, sax.wav

INTRODUCTION

Digital signal processing algorithms are often used to process continuous-time signals. It is necessary to convert a continuous-time signal into an equivalent discrete-time signal, apply the necessary digital signal processing algorithm to it, and then convert back the processed discrete-time signal into an equivalent continuous-time signal. In the ideal case, the conversion of a continuous-time signal into a discrete-time form is implemented by periodic sampling. To avoid aliasing the sampling frequency must be greater than or equal to twice the maximum frequency of the CT signal.

In this experiment, you will also be introduced to some simple discrete time signal manipulation – the downsampling, upsampling, and flip. Downsampling or decimation is a method of decreasing the sampling frequency. For example, y[n]=x[2n] is the algorithm for downsampling x[n] by a factor of 2. In this case the sampling frequency is halved. On the other hand, Upsampling is interpolation. When Upsampling is performed on x[n], it produces an approximation of the sequence that would have been obtained by sampling the CT signal at a higher rate. Upsampling may be carried out by adding another element, such as zero/s or the average of the adjacent elements, in every other element of the original samples. The flip command simply inverts the position of the samples either up/down or left/right. Addition and concatenation of two signals will also be a part of the experiment.

PROCEDURES

I. Simulation of Continuous Time Signal and the Sampling Process

All signals that you will be creating using computer are all discrete-time (DT). We can simulate a CT signal on a computer by simply increasing the number of samples per unit time (or the sampling frequency) at a significant value.

```
% Expt3_1
clf;
clear
%Simulation of CT Signal with F=10Hz
t = 0:0.0005:1;      % 0 to 1 sec
F = 10;
xa = cos(2*pi*F*t);
subplot(2,1,1)
plot(t,xa);grid
xlabel('Time, sec');ylabel('Amplitude');
title("Simulated Continuous-time signal\n Frequency=10Hz");

%Illustration of the Sampling Process to obtain DT Signal
 = 1/80;             %T is sampling period of 0.0125s
n = 0:T:1;
xn = cos(2*pi*F*n);
k = 0:length(n)-1;
subplot(2,1,2);
stem(k,xn);grid;
xlabel('Time index n');ylabel('Amplitude');
title("Discrete-time signal,x[n], after sampling");
```

STEP 1 Run the program Expt3_1 to generate and display both the continuous-time signal and its sampled version.

What is the frequency of the simulated CT signal, xa? 10 Hz

What is the sampling frequency to obtain the DT signal, xn? 20 Hz

What must be the minimum sampling frequency and the maximum sampling period to avoid aliasing? Fs (min) = 20 Hz T(max) = 0.05 sec

STEP 2 Run the same program for four other values of the sampling period listed below.

T=0.025 sec; T=0.05 sec; T=0.1sec

Which of the values will cause aliasing? T = 0.1 sec

State the sampling theorem: Sampling frequency is greater than or equal to the twice of maximum of the analog frequency. Fs ≥ 2Fmax

II. Downsampling, Upsampling, and Flip

Step 3 Create the vector v = [1 2 3 4 5].

What is the output after the execution of the following:

downsample(v,2) = ans = 1 3 5

upsample(v,2) = ans = 1 0 2 0 3 0 4 0 5 0

fliplr(v) = ans = 5 4 3 2 1

flipud(v) = ans = 1 2 3 4 5

STEP 4 The 'speech_dft.wav' contains a segment of a sampled signal. Play using any audio player so you can hear the original sound. Make a copy of this audio file in your Expt 3 folder.

Read and play the audio file in Octave using the commands shown below. Both x and fs are user-defined variables. x will store the content of the audio file and fs will store the sampling frequency.

```
>> [x,fs]=audioread('speech_dft.wav');
>> sound(x,fs);      % If fs is not included, the default sampling rate of 8kHz is used
```

Vector x contains the samples of the audio file. What type of vector is x? (row or column vector)

- Column

You can use the ***audioinfo*** command to obtain information about an audio file. Use this command and write the information that you were able to obtain about the 'speech_dft.wav'

Number of Channels: 1 Duration in seconds: 4.9902

Sampling Rate: 22050 Bits Per Sample: 16

Total Samples: 110033

STEP 5 Apply each of the following MATLAB functions on the same speech segment:

1. The MATLAB function **downsample** (use factor 2). Store the result on vector d.
2. The MATLAB function **upsample** (use factor 2). Store the result on vector u.
3. The MATLAB functions **fliplr** and **flipud**.

Describe what you hear when you played the signal in each case using the original fs. For example, sound(d,fs).

Downsample: The audio became faster.

upsample: The audio becomes slower.

fliplr: There is no any changes in the speed rate on the audio file.

flipud: The sound becomes reverse, it is not understandable.

What sampling frequency must you use to make d sound normal? 11,025

What sampling frequency must you use to make u sound normal? 44,100

Which of the four commands is non-invertible? The one with the fliplr command.

Create the corresponding .wav files for d and u using the new sampling frequency. Use the filenames 'speechDown.wav' and 'speechUp.wav' . Use the help command to look for the syntax of **audiowrite**. Verify if the created audio files sound almost the same as the original.

III. Concatenate and Mix

STEP 6 Concatenate means to link together in a chain or series. You can link two audio signals if both of them have the same number of channels, sampling frequency, and bits.

```
syntax: z=[A;B];    if A and B are column vectors
        z=[A,B];    if A and B are row vectors
```

Read sax.wav in A and flute.wav in B. Concatenate the two sounds, sax first and then followed by flute. Store the result in z. Describe what you hear. The sax is the first one to play and right after that the flute followed.

Write the first four elements of z:

```
First four elements of z:
0.032928
0.028320
0.020966
0.008423
```

1	0.032928
2	0.02832
3	0.020966
4	0.0084229

Write the last four elements of z:

```
Last four elements of z:
  -0.023071
  -0.023773
  -0.024841
  -0.025604
```

203111	-0.023071
203112	-0.023773
203113	-0.024841
203114	-0.025604

STEP 7 Adding or mixing sounds make use of the addition operation, that is, A+B. When you listen to the mixed sound, you will be hearing them playing simultaneously and clearly. Addition of signals is possible only if A and B have the same length, sampling frequency, bits, and size. To solve the problem on different lengths, you may add zeros at the end of the shorter file to make their lengths equal.

Assuming that A and B already have the same properties, add them and store the result using the variable mix. Describe what you hear. **They play at the same time.**

Write the first four elements of mix: 0.031097, 0.026917, 0.020142, 0.008057

```
First four elements of z:
0.031097
0.026917
0.020142
0.008057
```

Write the last four elements of mix: -0.010284, -0.006378, 0.000336, 0.006989

```
Last four elements of z:
-0.010284
-0.006378
0.000336
0.006989
```

List down the problem/s encountered and the solution/s made in adding the two audio files.

```
error: operator +: nonconformant arguments (op1 is 106739x1, op2 is 96375x1)
error: called from
    dsadfas at line 11 column 3
>> dsadfas
```

REMARKS and CONCLUSION:

In this experiment, I learned that the vector must be on the same size to be added together. Also I learned how to make the sound faster, slower, and even reverse by using the syntax upsample, downsample, and flip. I learned how to recreate or to manipulate the audio. Also I learned how to mix two audios together by adding zeros to the other file so it can have the same size.