3TP3 Lab 3 Report

Rui Qiu (400318681)

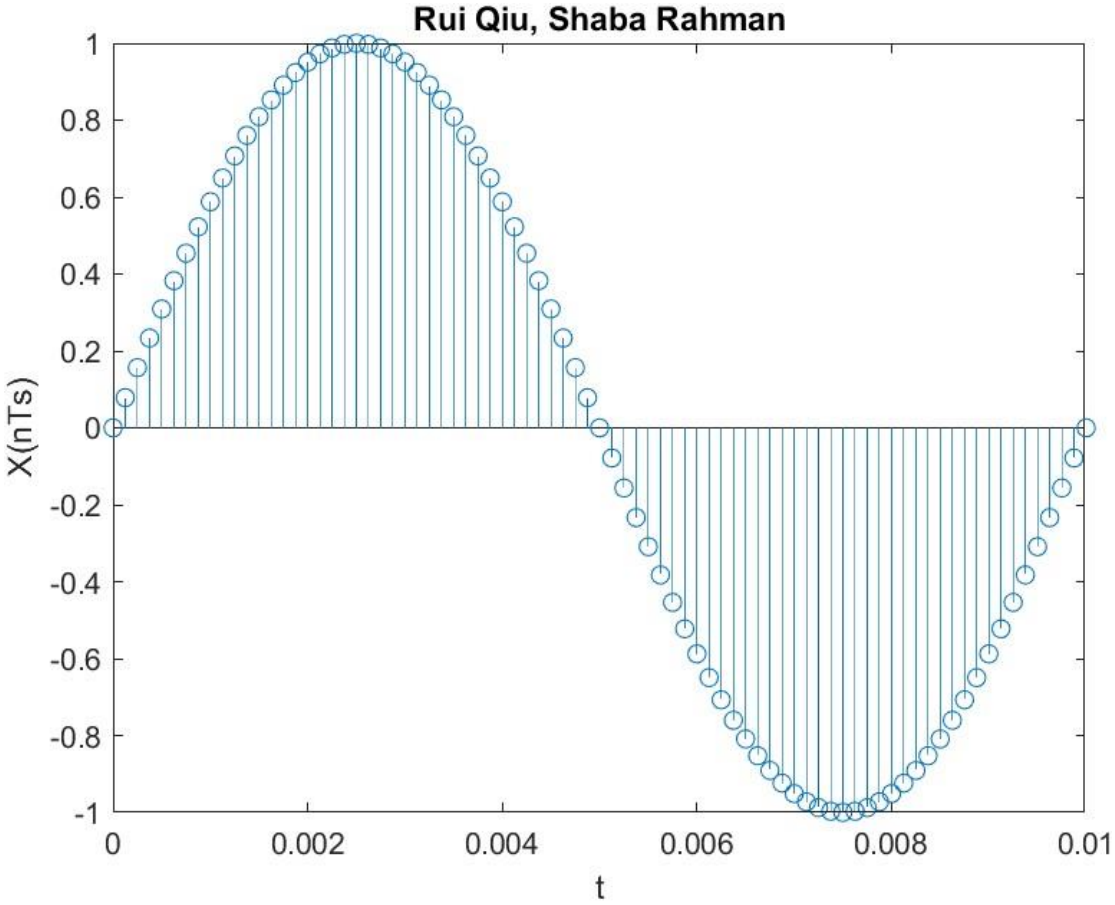
Shabab Rahman (400291441)

Instructor: Terry Todd

Section: T01

1:

Based on the code provided for Q1, the graph is obtained below. This is a discrete time sine signal that consists of impulse response every 125us seconds with the height of formula sin(2*pi*f*nplot) where nplot is the x value of each 125us increment. The signal is plotted using the stem command within the range of 0 to 10msec.



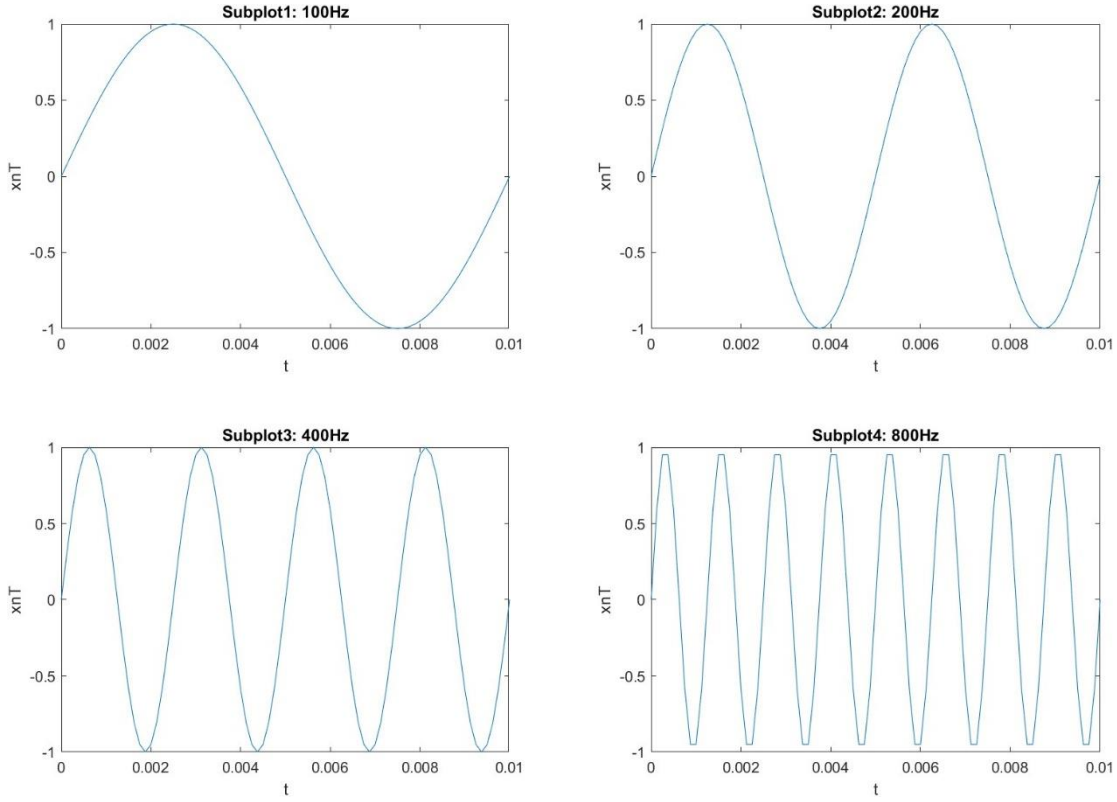
2:

Code:

```
clc;
clear;

% Use sinusoid frequency f = 300 Hz
f = [100,200,400,800];
%
% Sampling frequency and interval
fs = 8000;
Ts = 1/fs;
%
% Set time duration of plot, i.e., 10 msec.
tfinalplot = 10e-3;
%
% Make the time vector for the plot
nplot=0:Ts:tfinalplot;
```

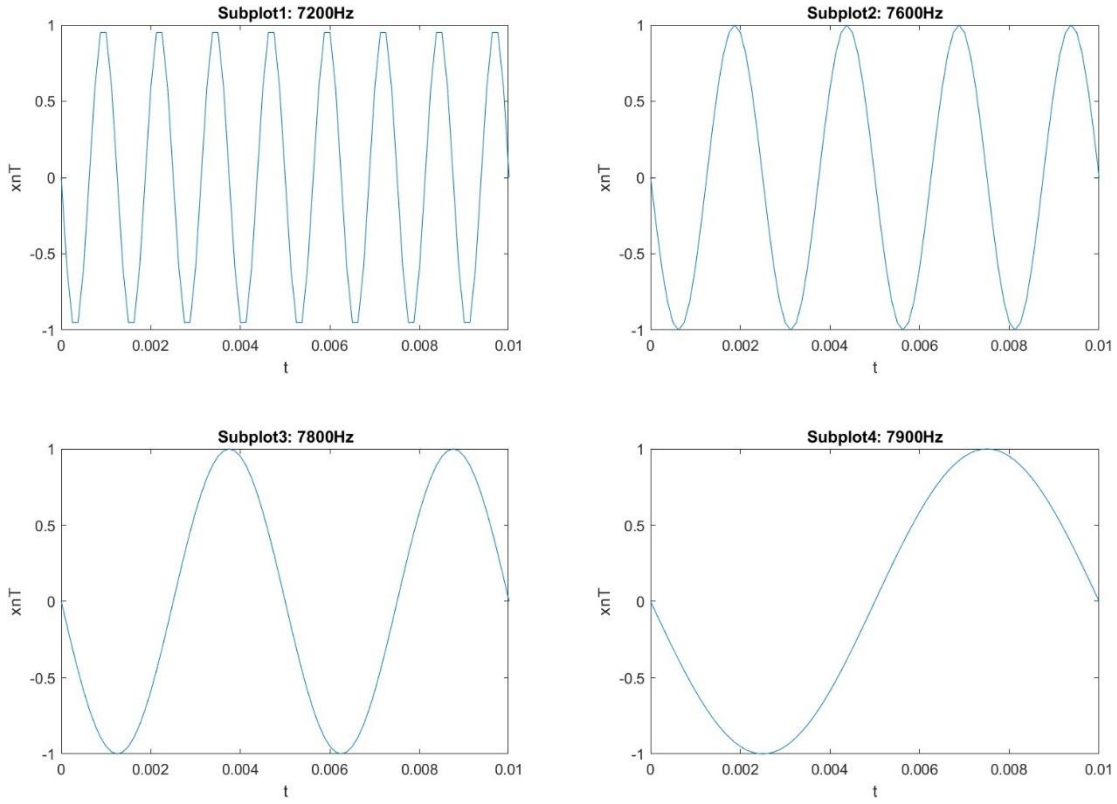
```
%
% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;
%
set(gcf, 'Position',  [400, 140, 1200, 800]);
%
C = [];
% Make the plot
for i = 1:4
    % Sample the sinusoid.
    xnT = sin(2*pi*f(1,i)*nsound);
    subplot(2,2,i);
    plot(nplot, xnT(1:length(nplot)));
    title("Subplot"+i+ ":"+" "+f(1,i)+"Hz");
    C = cat(2,C,xnT); %concatenate four xnT
    audiowrite(string(f(1,i))+' Hz.wav', xnT, fs);
end
exportgraphics(gcf, 'Q2.jpg');

audiowrite('concatenate Hz.wav', C, fs);
```

Graph:



Description of the concatenated four 2-second tone segments:
Based on the four frequencies provided, they are being concatenated from low frequency to high frequency accordingly in a single vector. During the 8-seconds sound file, the pitch of the sound is increasing accordingly after every two seconds. More specifically, with the increasing in input frequency, the output sound(pitch) is increasing as well.

3:



When the input frequencies are 7200, 7600, 7800 and 7900Hz and being concatenated from low frequency to high frequency in a single vector. The pitch of the sound file will be highest for the first two seconds and decreasing within every two seconds. With the increase of the input frequency, the output sound(pitch) is decreasing in this case.

And when keep increasing the input frequency beyond the sampling frequency, there is still possible to produce the sound, but it is inconsistent and do not match the same pattern of the input frequency. When the input frequencies are multiples of 4000, there will be no output sound.

The reason why this aliasing occurs is because the sampling rate frequency(8kHz) is not beyond two times of the input frequency(7200Hz). According to the Nyquist Sampling Theorem, a frequency component can be exactly represented by sample taken at a rate of at least two times the frequency component value. As a result, when the input frequency is beyond the one half the sampling frequency(4000Hz), aliasing will occur, and the output frequency will not follow the same pattern as the input

frequency.

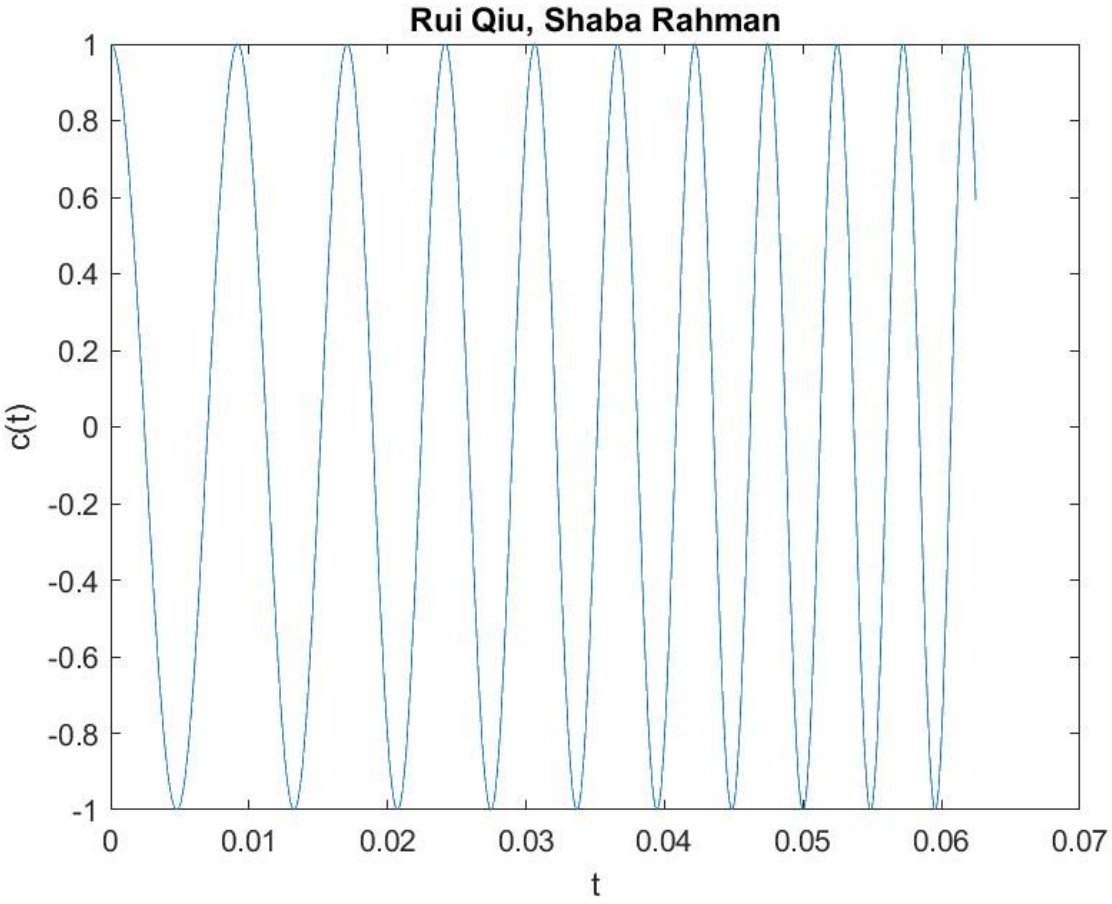
4: If the anti-aliasing pre-filtering is not used on the telephone system, when the energy of human voice beyond one half of the sampling frequency (the Nyquist sampling rate constraint is not observed), the aliasing will occur and the person who on the end of the phone will not understand what the other person is saying which will lose the most important functionality of the telephone. The anti-aliasing pre-filtering involves using a low-pass filter before ADC (Analog to Digital), it will reduce the frequency components above 3.5kHz, so that it will meet the Nyquist sampling rate constraint and there will be no significant aliasing occur. If this filtering is in place, then it will meet the Nyquist sampling rate, as a result the output frequency will follow the same pattern as the input frequency which with the increase of the input frequency, the output frequency will increase as well.

Aliasing of a Frequency Chirp Signal:

1. Code:

```
clc;
clear;
f = 100; %f1
u=2000;    %miu
fs = 32000; %sampling frequency
Ts = 1/fs;
% Make the time vector for the 2000 samples plot
nplot=0:Ts:Ts*1999;
% Make the time vector for replayed sound spurt
% Play the spurt for 8 seconds
tfinal = 8;
nsound=0:Ts:tfinal;
%sample the chirp signal
cT = cos(pi*u*(nsound.^2)+2*pi*f*nsound);
% Make the plot
plot(nplot, cT(1:length(nplot)));

% Save xnT as a wav sound file, soundfile.wav.
audiowrite('chirpsoundfile.wav', cT, fs);
exportgraphics(gcf, 'chirpsoundfile.jpg');
```

Graph:

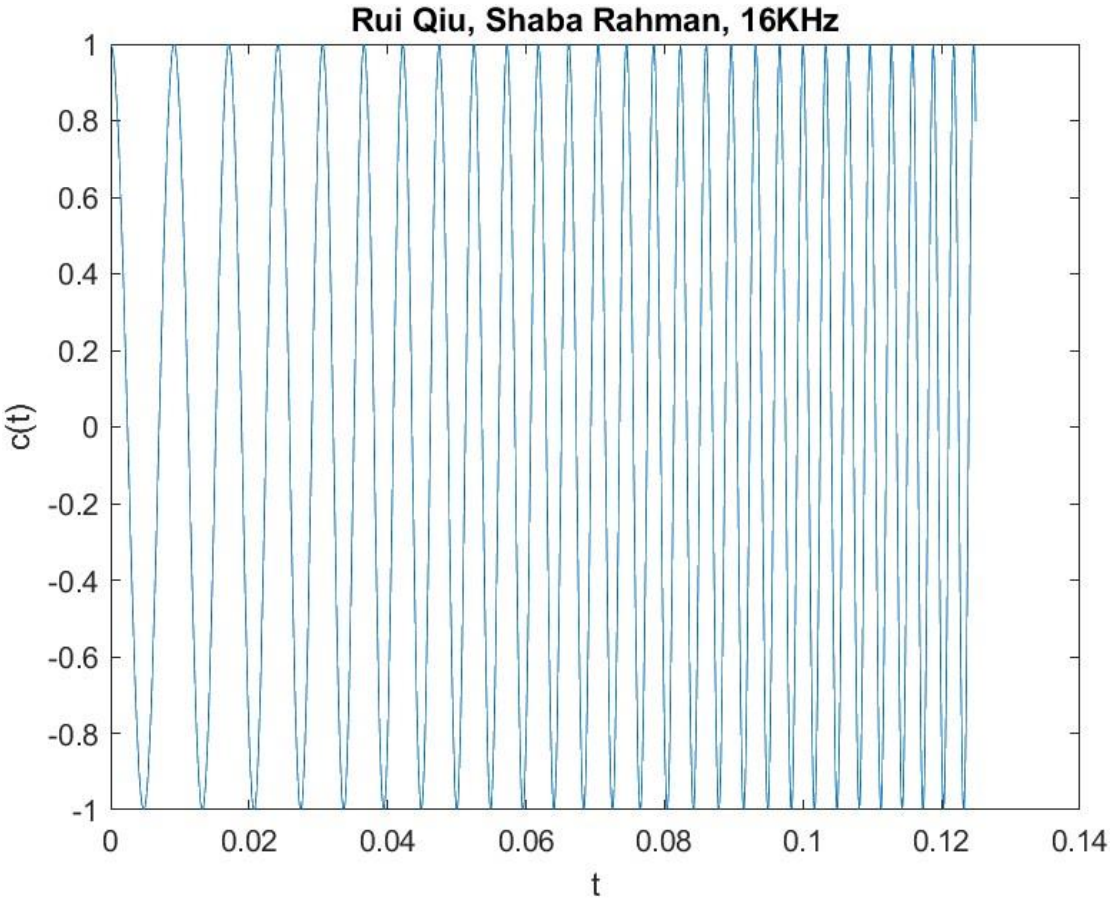


Based on the graph obtained, the sampling period will get decreasing as the sampling time increase. In conclusion, the frequency is increasing. The amplitude will remain unchanged at 1. Which it proves the theorem that by taking the derivative of the chirp signal that, the instantaneous frequency is given by f(t) = ut + f1, it indicates that the frequency is increasing linearly with time ut and staring at f1 Hz. It can also be observed that this linear function has a slope of u, which will be 2000 in this question. Therefore the frequency should start with a frequency of f1 = 100Hz and increase 2000Hz every second.

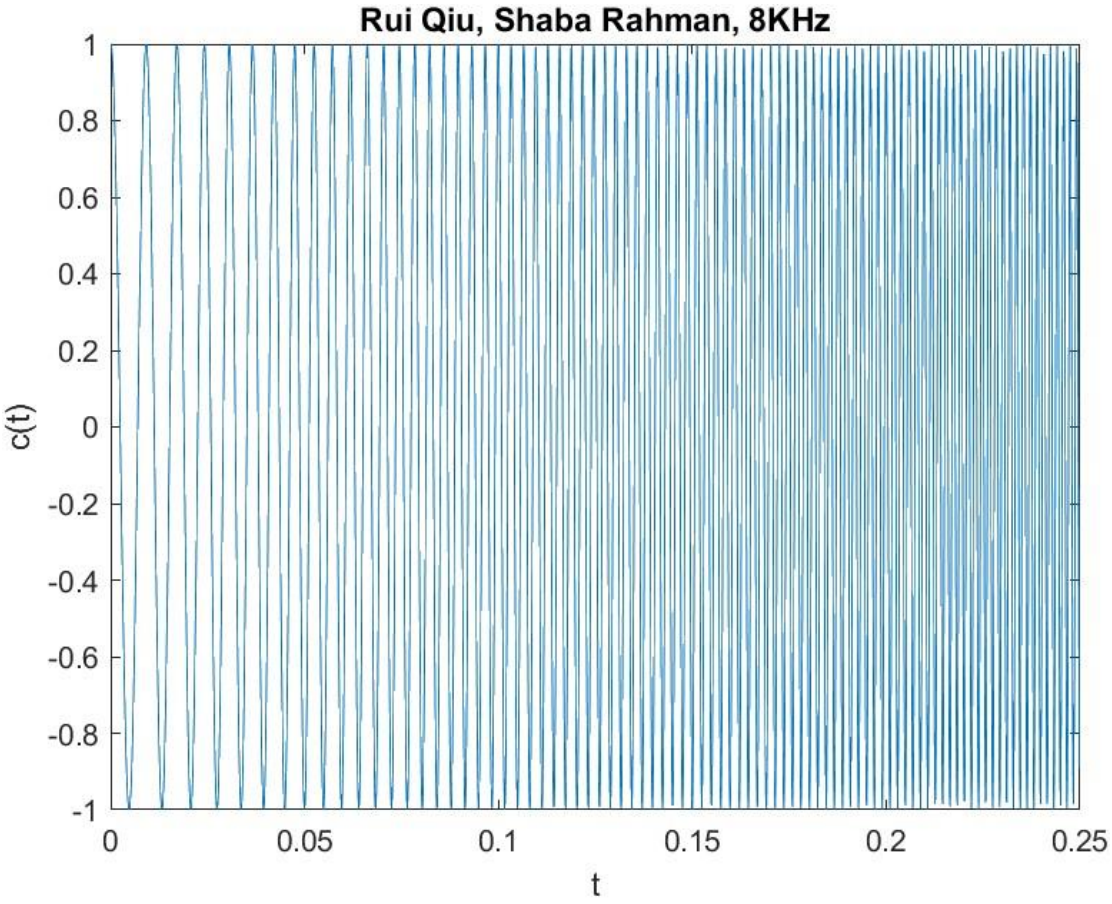
The audio file produced an 8 second frequency that starts with 100Hz and it increased 2kHz for every second, the sound increases from a low deep-pitch to a sharp high-pitch really quickly within the increase of time and reached to the approximate Nyquist constraint boundary 16000Hz to avoid aliasing when time is equal to eight.

2.

Plot for 16KHz:



Plot for 8KHz:



When setting the sampling frequency to 16KHz, the sound is increasing for the first four seconds and decrease for the last four seconds. And when setting the sampling frequency to 8KHz, the sound is increasing for the first two seconds then decreasing for the next two seconds and repeat the same pattern till the end. The reason why decreasing happens is due to the aliasing that the Nyquist sampling rate is not observed since the input frequency is greater than one half of the sampling frequency as time increases with the change of 2000Hz every second. For the 16KHz, the aliasing will occur at four seconds which 4*2000 + 100 > 16000/2; And for the 8KHz, the aliasing will occur at two second which 2*2000 + 100 > 8000/2. If an anti-aliasing filtering is being placed, for both two sampling frequencies 16KHz and 32Khz, as long as the filter can reduce the component frequency below one half of these frequencies, the output frequency should keep increasing till the end.

Experimenting the f1, fs and u values one at a time while keeping the others the same. When increasing the f1 value, the starting frequency will get increased and based on the derived formula, it will decrease the time that the input frequencies will reach the Nyquist sampling constraint rate and vice versa when there is no filtering applied. Based on experiments above, the change of Fs (sampling frequency) will change the time that the input frequency reaches the Nyquist sampling constraint rate boundary where it is important to determine the range of Fs given the information we obtained that the input frequency will increase 2kHz every second. The ideal Fs should be twice than the maximum input frequency to avoid aliasing. u as obtained above is the slope

of the linear chirp signal which behaves like a multiplier. It will determine how much frequency want to increase for every second. It has a significant impact for the chirp signal. It should be very cautious to choose a valid u to avoid too quick frequency increment that causing aliasing.